Apgūsti robotikas pamatus

"SumoBoy" v 1.0

ROBOT
NEST

Mācību materiāls un darba lapas izmantojamas kopā ar komplektu robotikas pamatu apgūšanai "SumoBoy".

Ierosinājumus un ieteikumus, kas radušies komplekta izmantošanas procesā, lūdzam sūtīt karlis@robot-nest.com

Lūdzam sekot mācību materiāla atjaunotajām versijām mūsu interneta vietnē www.robot-nest.com

Mācību materiāls sagatavots RTU Izdevniecībā

Redaktores Anita Vēciņa, Lilita Vīksna
Datordizainere Jekaterina Lukina
Foto Elīna Karaseva

SATURS

IEVADS

Mācību komplekts robotikas pamatu apgūšanai "SumoBoy" paredzēts vienam mācību gadam (pēc skolas izvēles 7.–12. klasē). Nodarbības notiek vienu reizi nedēļā, katras nodarbības ilgums trīs akadēmiskās stundas. Pēc robotikas pamatu apgūšanas skolēni var izmantot šo komplektu arī turpmāk, lai pilnveidotu robotu sumo sacensībām, kā arī lai izmēģinātu spēkus modernās robotikas jomā. Komplektu papildina mācību programma un darba lapas katrai stundai. Mācību programmu var realizēt kā atsevišķu mācību priekšmetu vai kā interešu izglītības nodarbības.

Mācību gada laikā skolēni, pakāpeniski veicot aizvien sarežģītākus uzdevumus, apgūst tos elektronikas elementus, kas nepieciešami robotu konstruēšanā, arī līdzstrāvas dzinēju vadību un ar sensoriem iegūto datu apstrādi. Atsevišķs uzdevumu bloks paredzēts robotu programmēšanas pamatu apgūšanai. Mācību komplektā ietvertais robots "SumoBoy" atbilst minisumo robotu starptautisko sacensību noteikumiem. Tādējādi skolēni varēs pilnvērtīgi piedalīties Latvijas un starptautiska līmeņa sacensībās, aizstāvot savas skolas vai novada godu.

Programmas mērķis

Veicināt skolēnu interesi par robotiku kā vienu no straujāk augošām inženierzinātņu jomām; motivēt skolēnus saistīt savu tālāko izglītību ar inženierzinātnēm.

Programmas uzdevumi

Apgūt prasmi kombinēt elektrotehnikas, elektronikas un datorzinību elementus, lai, izmantojot mācību komplektā esošās sastāvdaļas, varētu izveidot robotu, kas veic programmatūrā iestrādātās darbības.

Skolēniem vispārizglītojošajās skolās fizikas programmā nav paredzēta lodēšanas apguve, tādēļ šeit ietvertajā robotikas pamatu programmā tam paredzēta viena papildu nodarbība. Darba lapās ir pievienoti uzdevumi drošības tehnikas jautājumu un multimetra lietošanas apguvei. Mācību gada noslēgumā tiek rīkotas sumo robotu sacensības, lai skolēniem būtu motivācija mācīties.

Sagaidāmie rezultāti, apgūstot programmu:

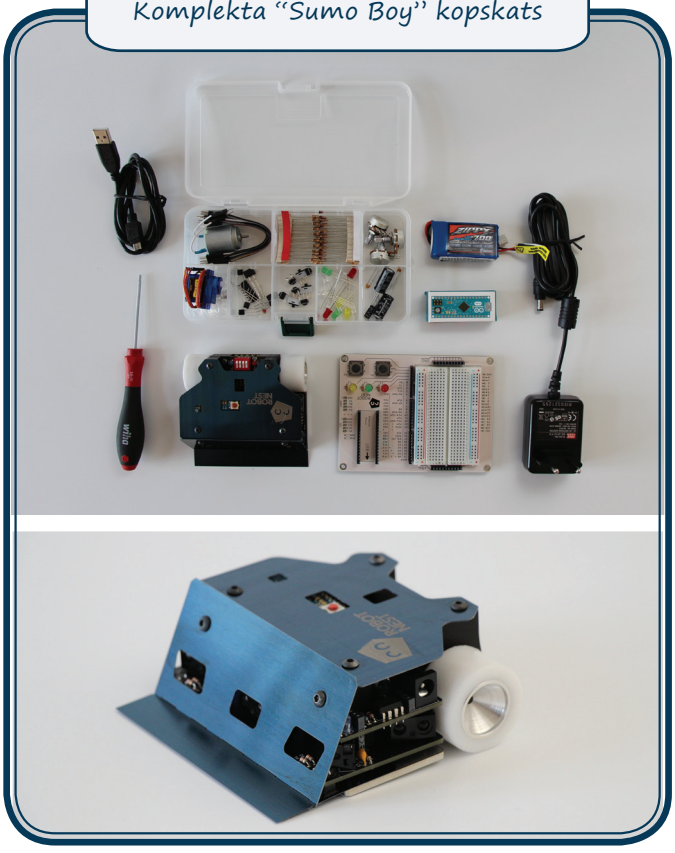
- Apgūti drošības tehnikas un lodēšanas pamati, ir iemaņas multimetra lietošanā.
- Gūts priekšstats par elektronikas pamata elementiem un to lietojumu.
- Gūts priekšstats par elektrodzinēju vadību un sensoru datu apstrādi.
- Iegūtas pamatiemaņas programmēšanā.

Mācību materiāla uzbūve

Mācību materiāls ir sadalīts tematos, kas aplūko kādu konkrētu robota uzbūves aspektu, piemēram, elektronikas elementus, motoru vadību u. c. Katram tematam ir sagatavots vispārīgs apraksts un darba uzdevumi. Temati tiek numurēti pēc kārtas visā mācību materiālā, bet darba lapas katrā tematā, t. i., katra temata patstāvīgie darbi sākas ar 1. darba lapu.

Darba lapas veidotas tā, lai tās varētu izdrukāt un turēt sev līdzās kā "špikeri" attiecīgā uzdevuma izpildes laikā.

Komplekta "Sumo Boy" kopskats



1. ELEKTRONIKAS ELEMENTI

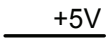
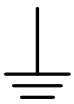
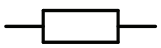
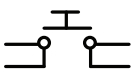
1.1. Mērķis

Temata mērķis ir iepazīstināt ar konstruktora prototipēšanas plates elementiem un iespējām, kā arī sniegt pamatzināšanas par elektrisko ķēžu elementiem.

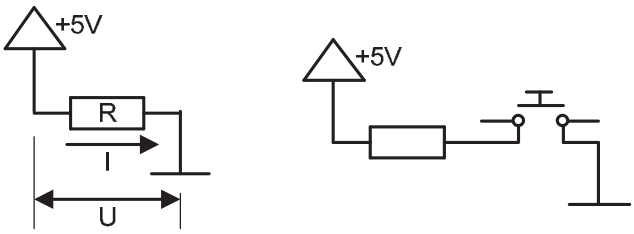
1.2. Teorētiskā daļa

Vienkārša elektriskā ķēde

Robots sastāv no mehāniskās, elektroniskās un programmatūras daļas. Šajā nodaļā tiek aplūkota elektroniskā daļa — elektriskās ķēdes. Jebkura elektriskā ķēde sastāv no enerģijas avota (šajā konstruktorā tiek izmantots akumulators), vadītāja (vadiem) un pretestības, kas patērē strāvu, lai veiktu darbu — veiktu aprēķinus, darbinātu motorus vai nolasītu sensoru datus. Shematiski šos elementus apzīmē šādi:

Apzīmējums		Shēmas simbols
		Enerģijas avota (baterijas, akumulatora) pozitīvā pieslēgvieta. Elektriskā strāva ķēdē plūst no pozitīvā uz negatīvo pieslēgumu jeb polu.
		Enerģijas avota (baterijas, akumulatora) negatīvā pieslēgvieta vai kopējā pieslēgvieta. Dažkārt to sauc arī par zemi vai masu (automašīnu ķēdēm).
		Pretestība jeb rezistors.
		Vads jeb vadītājs bez nozīmīgas pretestības.
		Poga.
		Viena un divu polu slēdzis.

Vienkāršas ķēdes piemēri:



Termini

- Par **pretestību** sauc vadītāja pretdarbību strāvas plūsmai. To mēra omos (Ω). Pretestību apzīmē ar ***R***.
- Par **strāvas stiprumu** sauc daļiņu plūsmu, kas var pārnest elektrisko lādiņu. Tās stiprumu mēra ampēros (A). To var salīdzināt ar ūdens plūsmu caurulē — jo vairāk pagriež krānu, jo stiprāka ūdens plūsma. Strāvu apzīmē ar ***I***. Strāva plūst no pozitīvās ķēdes pieslēgvietas jeb pola uz negatīvo polu — tā, kā tas norādīts piemēra ķēdēs.
- Par **spriegumu** sauc elektriskā potenciāla starpību. To mēra voltos (V) un apzīmē ar ***U***.

Minētos trīs elementus saista sakarība, kas zināma ar nosaukumu **Oma likums**:

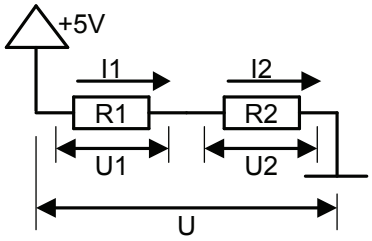
$$I = \frac{U}{R} \quad \text{jeb} \quad U = IR$$

Zināšanai. *Šis likums ir nosaukts vācu zinātnieka Georga Simona Oma vārdā, kurš to formulēja 1827. gadā, pierādot, ka elektriskajā ķēdē pastāv sakarība starp ķēdes posma spriegumu un strāvu.*

Virknes un paralēlais slēgums

Dažkārt sarežģītākās ķēdēs ir nepieciešams izveidot ķēdi ar vairākiem rezistoriem, kuri ir slēgti viens aiz otra — virknes slēgums vai viens otram blakus — paralēlais slēgums:

Virknes slēgums



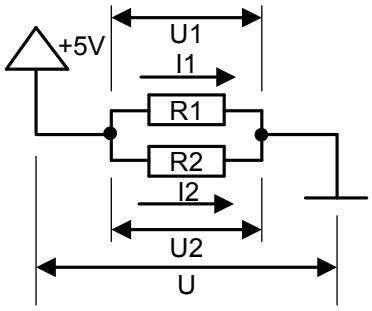
$$U_1 = I_1 R_1 \qquad U_2 = I_2 R_2 \qquad U = IR$$

$$U = U_1 + U_2$$

$$R = R_1 + R_2$$

$$I = I_1 = I_2$$

Paralēlais slēgums



$$U_1 = I_1 R_1 \qquad U_2 = I_2 R_2 \qquad U = IR$$

$$U = U_1 = U_2$$

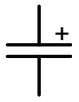
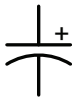
$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

$$I = I_1 + I_2$$

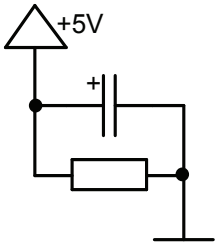
Kondensators

Lai panāktu elektrisko un elektronisko ķēžu darbību sev vēlamā veidā, viens no būtiskiem elementiem ir kondensators, kura galvenais uzdevums ir uzkrāt lādiņu. Tādējādi tā darbība ir līdzīga akumulatoram, kurā uz laiku var uzglabāt zināmu lādiņu un tad, kad tas ir nepieciešams, izmantot to elektriskajā ķēdē. Šajā tematā aplūko tikai elektrolītisko kondensatoru, kas parasti sastāv no vismaz divu slāņu vadītājiem, kuriem pievienoti izvadi pieslēgšanai elektriskajā ķēdē, un dielektriķi (no elektrisko strāvu nevadoša materiāla jeb izolatori).

Kondensatora shematisks apzīmējums:

Apzīmējums		Shēmas simbols
		Elektrolītiskā kondensatora dažādi apzīmējumi, kuros ar "+" norāda uz ķēdes pozitīvo pieslēgvietas virzienu. Kā redzams, arī kondensatora attēlojums skaidri norāda uz divām vadītāja virsmām, kas atdalītas ar nevadošu atstarpi — izolatoru.

Vienkāršas shēmas piemērs:



Šis ir iepriekš dotās shēmas papildinājums. Tas nodrošina, ka elektriskā strāvā neplūdīs caur pretestību, iekams nebūs uzlādējies kondensators, t. i., strāvas plūšana tiek uz īsu brīdi aizkavēta.

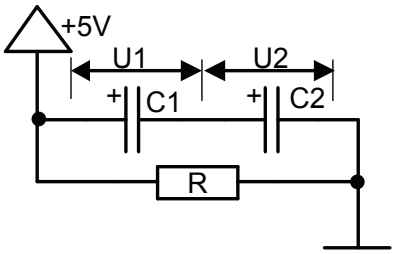
Termini

- Lādiņa daudzumu, kuru var uzkrāt kondensators, sauc par tā **kapacitāti** un apzīmē ar ***C***. Jo lielāks vadītāju kopējais laukums, jo lielāka kondensatora kapacitāte.
- Kapacitātes mērvienība ir **farads** (**F**). Parasti lietojamo kondensatoru kapacitāte ir no dažiem pikofaradiem (pF) līdz desmitiem un simtiem tūkstošu mikrofaradu (µF). Dažos īpašos pielietojumos tiek izmantoti sevišķi lielas ietilpības kondensatori, kur kapacitāte var sasniegt vairākus faradus.
- Kondensatoru raksturo arī tā **darba spriegums** (**V**), kurā tā īpašības paliek nemainīgas. Ieteicams izmantot mazāku spriegumu, jo lielāks spriegums var kaitēt kondensatoram.

Zināšanai. *Pirmo kondensatoru 1745. gadā izgatavoja vācu fiziķis Evalds Jurgens fon Kleists un holandiešu fiziķis Pīters van Mušenbruks. Tas notika Leidenē un izgatavotā ierīce sastāvēja no stikla trauka, kam iekšpusē un ārpusē bija folijas klājumi, to nosauca par "Leidenes trauku". Šo kondensatoru varēja uzlādēt ar elektrostatisko mašīnu, un tas spēja uzkrāt visai ievērojamu augstsprieguma lādiņu.*

Līdzīgi kā ar pretestībām, sarežģītākās ķēdēs ir iespējams izveidot slēgumu ar vairākiem kondensatoriem, kuri ir slēgti viens aiz otra (virknes slēgumā) vai viens otram blakus (paralēlā slēgumā):

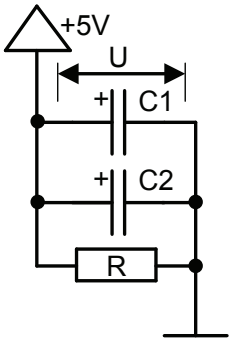
Virknes slēgums



$$\frac{1}{C} = \frac{1}{C_1} + \frac{1}{C_2}$$

Ļauj pielāgot kapacitāti konkrētām vajadzībām.

Paralēlais slēgums



$$C = C_1 + C_2$$

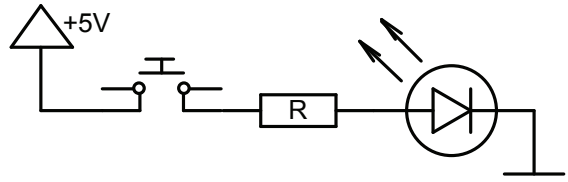
Ļauj palielināt kopējo kapacitāti.

Mirdzdiode

Lai praktiskajos uzdevumos attēlotās shēmas būtu saprotamas, tiek ieviests papildu elements – mirdzdiode, kas darbojas līdzīgi spuldzītei un vienlaicīgi nodrošina strāvas plūšanu tikai vienā virziena, t. i., atšķirībā no spuldzītes tā nedarbosies, ja nebūs pieslēgta elektriskajai ķēdei pareizā virzienā. Sīkāk par diodēm un citiem līdzīgiem elementiem pastāstīts tematā "Pusvadītāji". Mirdzdiodes shematiskais apzīmējums ir šāds:

Apzīmējums	Shēmas simbols
	Mirdzdiode; bultiņa tās simbolā norāda uz strāvas plūšanas virzienu.

Vienkāršas shēmas piemērs:



Šāda shēma ļauj izveidot ķēdi, kas nodrošina mirdzdiodes spīdēšanu tad, kad tiek nospiesta poga.

Jāuzsver, ka mirdzdiode darbosies tikai tad, ja tās pozitīvā un negatīvā pieslēgvieta būs pieslēgta atbilstoši strāvas plūšanas virzienam tā, kā tas norādīts shēmā. Mirdzdiodei ir salīdzinoši maza pretestība, tādēļ, lai shēma darbotos pareizi, tai ir jāpievieno pretestība tieši pirms vai pēc mirdzdiodes.

1.1. DARBA LAPA. Mirdzdiodes ieslēgšana

Mērķis

Izveidot savu pirmo elektrisko ķēdi, kura dota teorētiskajā daļā.

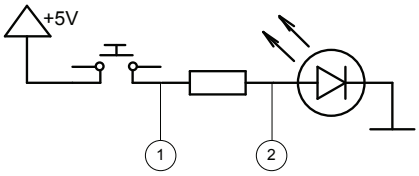
Darba izpildes soļi

1. Izveidot shēmu, kas redzama attēlā.
2. Pieslēgt shēmu strāvas avotam un nospiest pievienoto pogu. Lai top gaisma!
3. Atslēgt shēmu no strāvas avota.
4. Izmantot multimetru (testeri) pretestības mērīšanai starp punktiem **1** un **2**. Cik liela ir pretestība omos?
5. Pieslēgt shēmu strāvas avotam.
6. Nofiksēt pogu nospiestā veidā vai aizvietot to ar vadu tā, lai mirdzdiode nepārtraukti spīd.
7. Izmantot multimetru (testeri) sprieguma mērīšanai starp punktiem **1** un **2**. Cik liels ir spriegums voltos?

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Pretestība (330 Ω)	1 gab.
Mirdzdiode	1 gab.
Poga	1 gab.
Montāžas vads	4 gab.

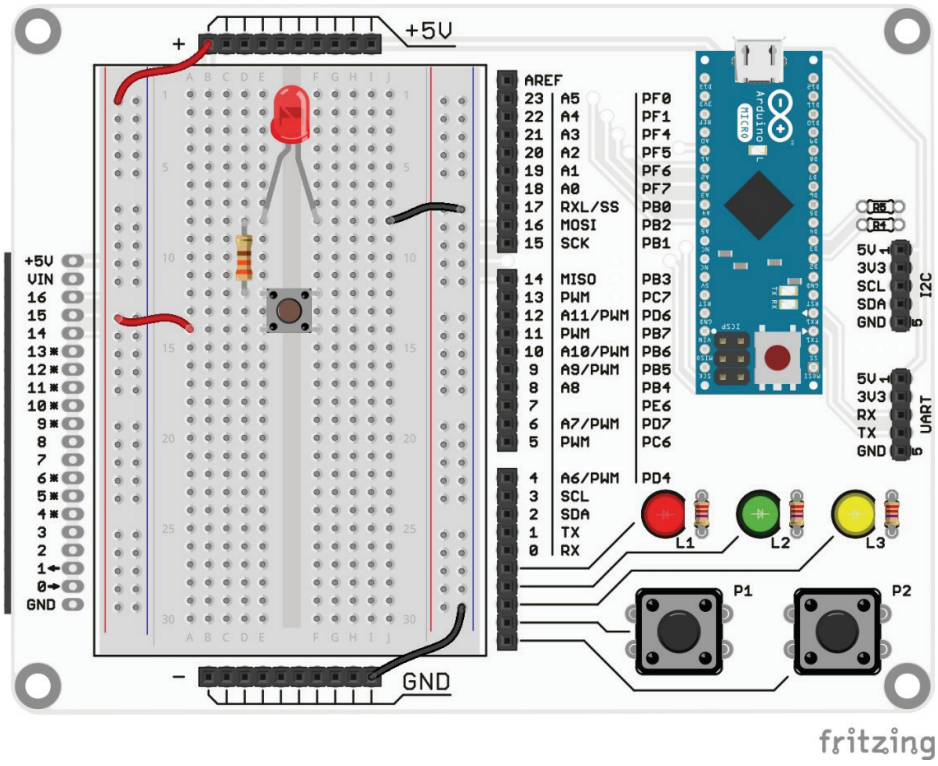
Izveidojamā shēma



Šāda shēma ļauj izveidot ķēdi, kas nodrošina mirdzdiodes spīdēšanu tad, kad tiek nospiesta poga.

Shēmā ir divi mērījumu punkti **1** un **2**, kas paredzēti sprieguma (***U***) un pretestības (***R***) mērīšanai.

Shēma



1.2. DARBA LAPA. Mirdzdiodes pārslēgšana

Mērķis

Iemācīties lietot slēdžus interesantiem uzdevumiem.

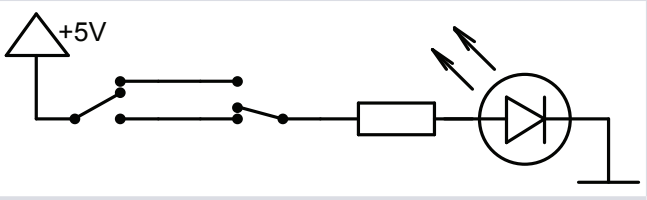
Darba izpildes soļi

1. Izveidot shēmu, kas redzama attēlā.
2. Pieslēgt shēmu strāvas avotam un ieslēgt vienu vai otru slēdzi.

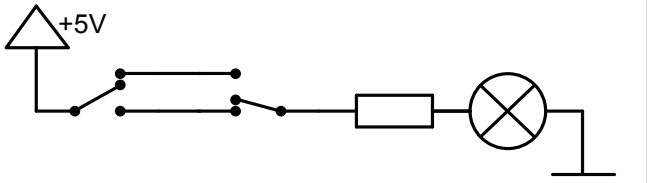
Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Pretestība (330 Ω)	1 gab.
Mirdzdiode	1 gab.
Montāžas vads	6 gab.
Slēdzis	2 gab.

Izveidojamās shēmas

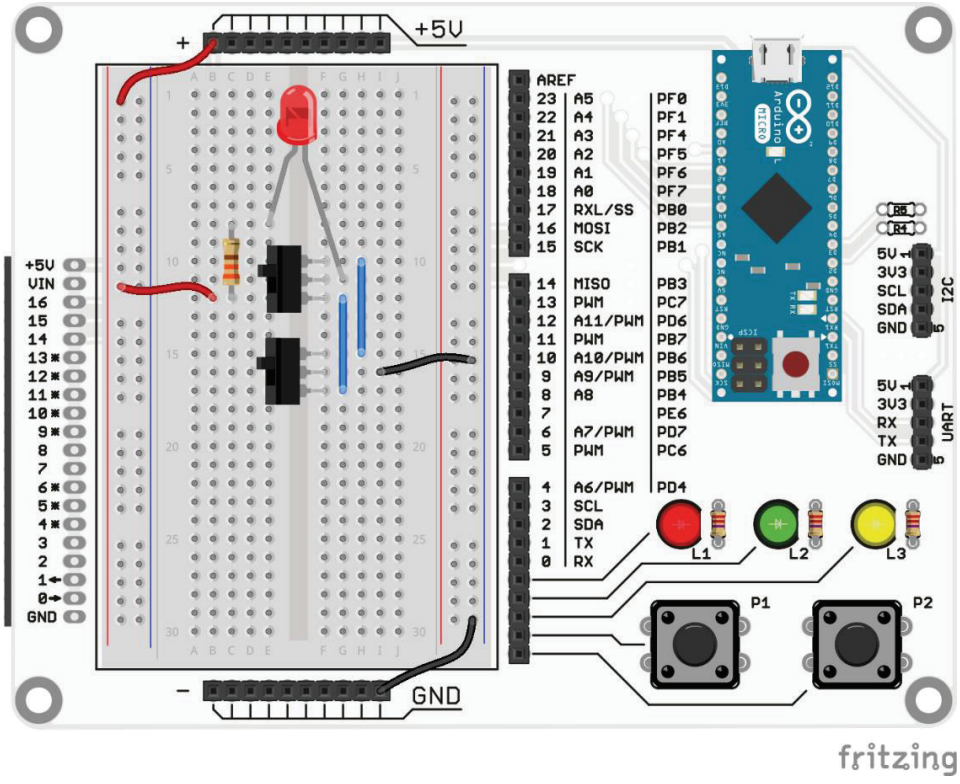


Kāpņu telpu slēdzis, kur gaismu var ieslēgt un izslēgt no vairākām vietām. Mirdzdiodes var aizstāt ar vienu spuldzīti. Tā, kā tas parādīts zemāk.



Ir jāuzsver, ka parasti spuldzīte ir vienlaicīgi arī pretestība. Tādēļ no shēmas pretestību var izņemt.

Shēma



1.3. DARBA LAPA. Pretestību virknes slēgums

Mērķis

Iemācīties izmantot pretestību virknes slēgumu.

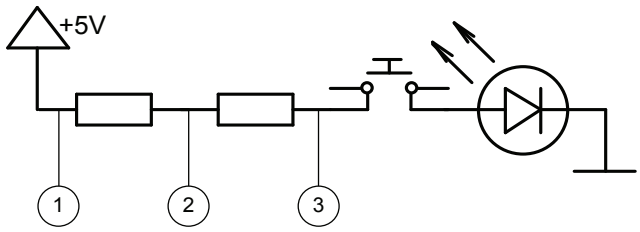
Darba izpildes soļi

1. Izveidot shēmu, kas redzama attēlā.
2. Pieslēgt shēmu strāvas avotam un ieslēgt pogu.
3. Atslēgt shēmu no strāvas avota.
4. Izmantot multimetru (testeri) pretestības mērīšanai starp punktiem **1** un **2**. Cik liela ir pretestība omos?
5. Izmantot multimetru (testeri) pretestības mērīšanai starp punktiem **2** un **3**. Cik liela ir pretestība omos?
6. Izmantot multimetru (testeri) pretestības mērīšanai starp punktiem **1** un **3**. Cik liela ir pretestība omos?
7. Pārbaudīt, vai izmērītā pretestība atbilst aprēķinātajai, izmantojot virknes slēguma pretestības aprēķinu.

Nepieciešamie materiāli

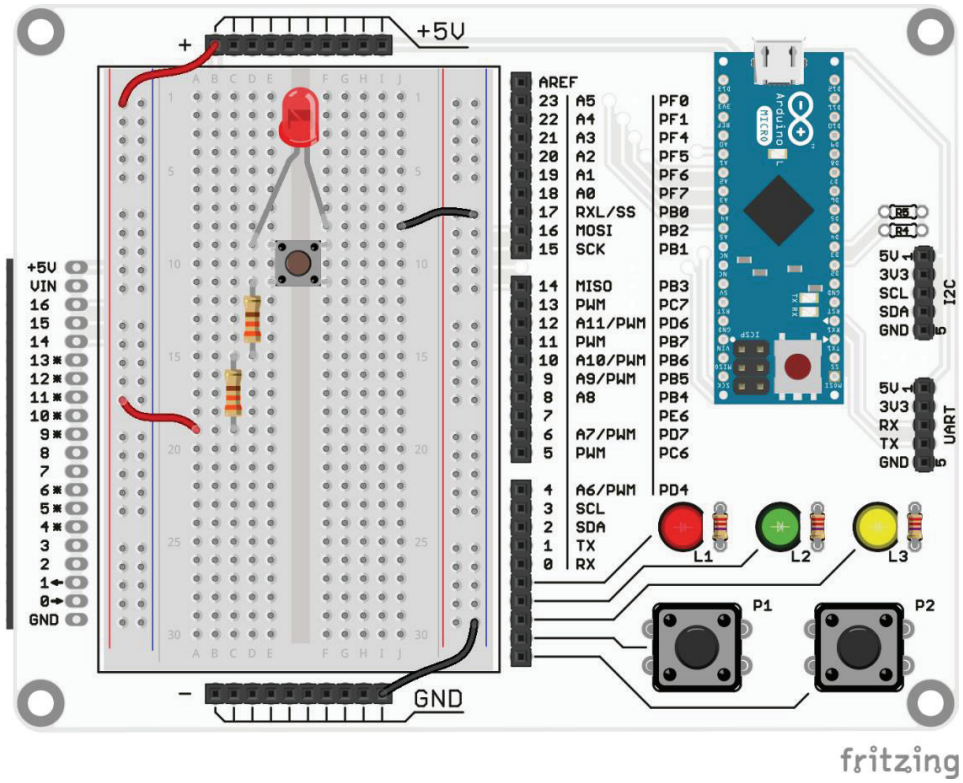
Materiāls/detaļa	Skaits
Pretestība (330 Ω)	2 gab.
Mirdzdiode	1 gab.
Poga	1 gab.
Montāžas vads	4 gab.

Izveidojamā shēma



Iepriekš aplūkotās mirdzdiodes shēma, kas papildināta ar vēl vienu pretestību, veidojot pretestību virknes slēgumu.

Shēma



1.4. DARBA LAPA. Pretestību paralēlais slēgums

Mērķis

Iemācīties izmantot pretestību paralēlo slēgumu.

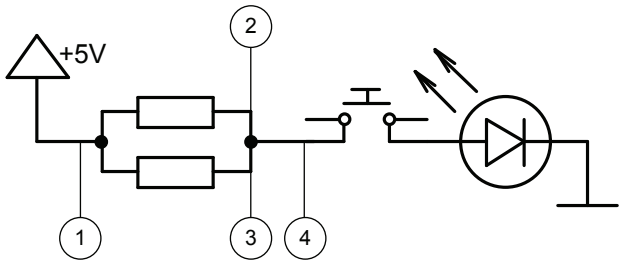
Darba izpildes soļi

1. Izveidot shēmu, kas redzams attēlā.
2. Pieslēgt shēmu strāvas avotam un ieslēgt pogu.
3. Atslēgt shēmu no strāvas avota.
4. Izmantot multimetru (testeri) pretestības mērīšanai starp punktiem **1** un **2**. Cik liela ir pretestība omos?
5. Izmantot multimetru (testeri) pretestības mērīšanai starp punktiem **1** un **3**. Cik liela ir pretestība omos?
6. Izmantot multimetru (testeri) pretestības mērīšanai starp punktiem **1** un **4**. Cik liela ir pretestība omos?
7. Pārbaudīt, vai izmērītā pretestība atbilst aprēķinātajai, izmantojot paralēlā slēguma pretestības aprēķinu.

Nepieciešamie materiāli

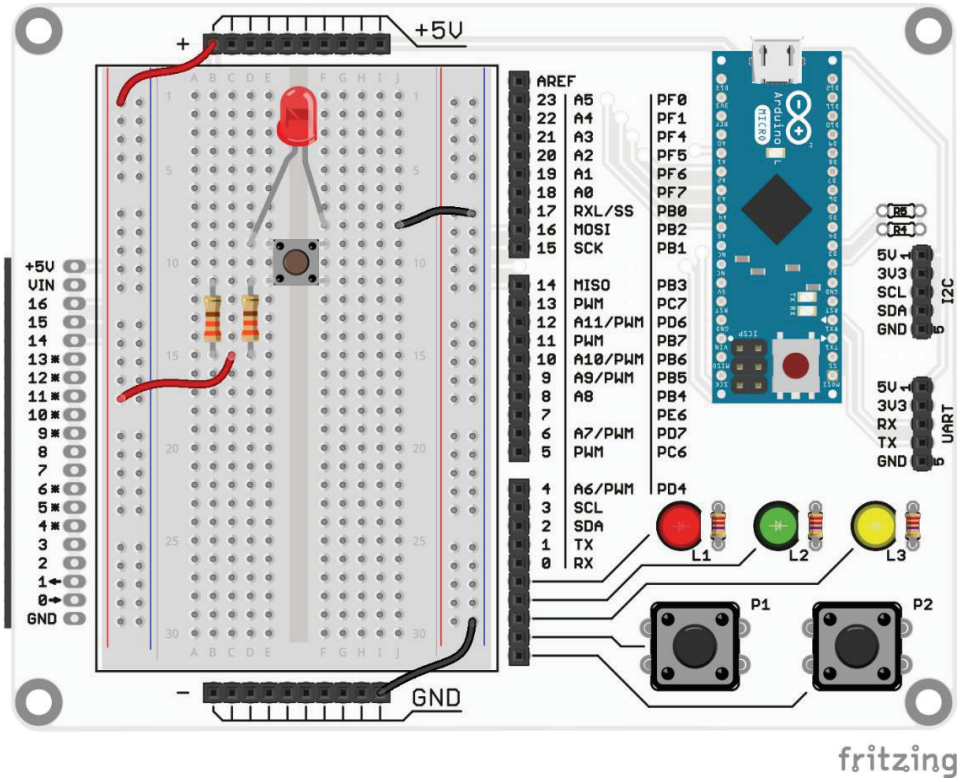
Materiāls/detaļa	Skaits
Pretestība (330 Ω)	2 gab.
Mirdzdiode	1 gab.
Poga	1 gab.
Montāžas vads	4 gab.

Izveidojamā shēma



Iepriekš aplūkotās mirdzdiodes shēma, kas papildināta ar vēl vienu pretestību, veidojot pretestību paralēlo slēgumu.

Shēma

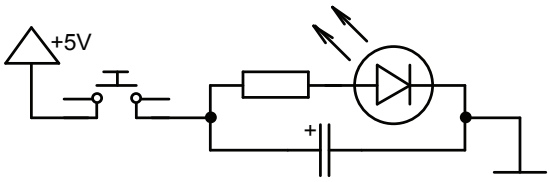


1.5. DARBA LAPA. Mirdzdiode ar kondensatoru

Mērķis

Iemācīties izmantot kondensatorus.

Izveidojamā shēma



Shēma nodrošina vienmērīgu mirdzdiodes izdzišanu pēc strāvas atslēgšanas, t. i., atslēdzot strāvu, kondensators nodrošina nepieciešamo enerģiju mirdzdiodes darbībai noteiktam laika sprīdim.

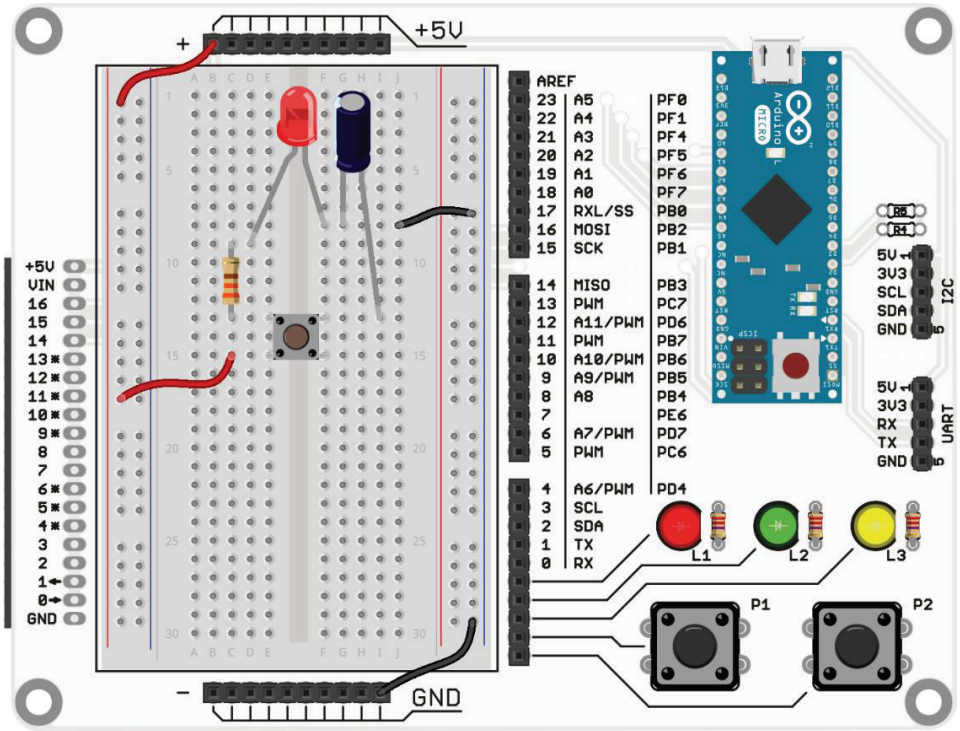
Darba izpildes soļi

1. Izveidot shēmu, kas redzama attēlā.
2. Pieslēgt shēmu strāvas avotam un nospiest pogu. Cik ilgs laiks sekundēs pagāja, kopš iedegās mirdzdiode?

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Pretestība (330 Ω)	1 gab.
Mirdzdiode	1 gab.
Elektrolītiskais kondensators (2200 μF, 10 V)	1 gab.
Poga	1 gab.
Montāžas vads	2 gab.

Shēma



1.6. DARBA LAPA. Kondensatoru virknes un paralēlais slēgums

Mērķis

Iemācīties izmantot kondensatoru paralēlo un virknes slēgumus.

Darba izpildes soļi

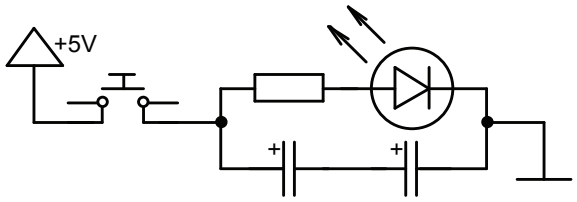
1. Izveidot 1. shēmu, kas redzama attēlā.
2. Pieslēgt shēmu strāvas avotam un nospiest pogu. Cik ilgs laiks sekundēs pagāja, kopš iedegās mirdzdiode?
3. Izveidot 2. shēmu, kas redzama attēlā.
4. Pieslēgt shēmu strāvas avotam un nospiest pogu. Cik ilgs laiks sekundēs pagāja, kopš iedegās mirdzdiode?
5. Kā mainījās laiks virknes slēguma shēmai?

Nepieciešamie materiāli

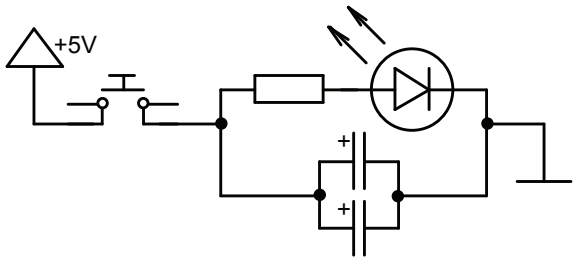
Materiāls/detaļa	Skaits
Pretestība (330 Ω)	1 gab.
Mirdzdiode	1 gab.
Elektrolītiskais kondensators (2200 µF, 10 V)	2 gab.
Poga	1 gab.
Montāžas vads	2 gab.

Izveidojamās shēmas

Virknes slēgums

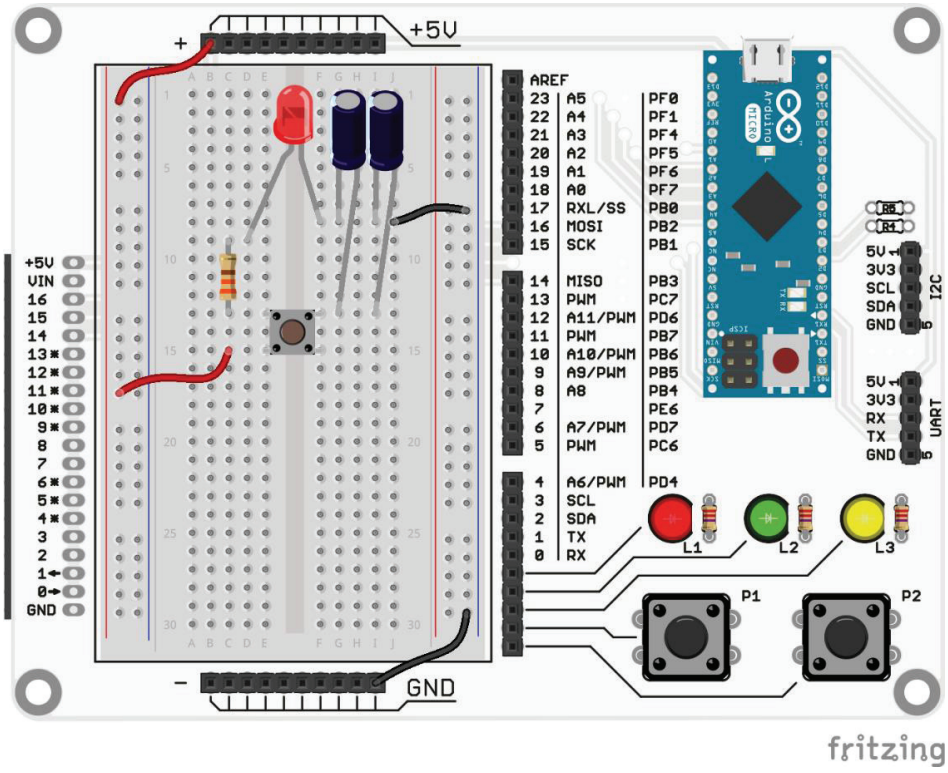


Paralēlais slēgums

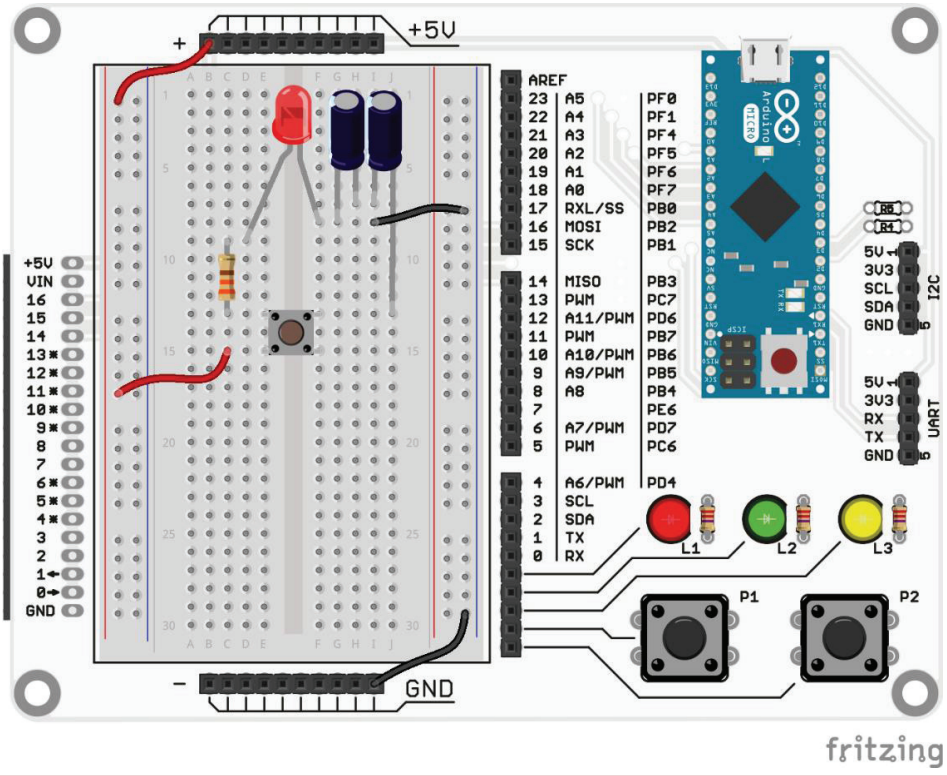


Shēmas nodrošina nelielu laika sprīdi, kurā mirdzdiode vienmērīgi nodziest, t. i., atslēdzot strāvu, kondensatori nodrošina nepieciešamo enerģiju mirdzdiodes darbībai noteiktam laika sprīdim. Atkarībā no slēguma veida laika aizture ir mainīga.

Paralēlā slēguma shēma



Virknes slēguma shēma



2. PROGRAMMĒŠANAS VIDES SAGATAVOŠANA

2.1. Mērķis

Temata mērķis ir soli pa solim palīdzēt izveidot programmēšanas vidi, konfigurēt to un izveidot savu pirmo programmu, izmantojot komplektā ietverto programmējamo mikrokontrolleri Arduino Micro®.

2.2. Teorētiskā daļa

Vispārīgs apraksts

Mācību konstruktorā ir iekļauts Arduino Micro tipa programmējams mikrokontrolleris, kas palīdzēs apgūt programmēšanas pamatus, kā arī sagatavos jūs pilnvērtīgai minisumo robota programmēšanai un sagatavošanai sacensībām. Mikrokontrolleris ir balstīts uz *ATmega32u4* mikroprocesora bāzes sadarbībā ar Itālijas uzņēmumu "Adafruit". Tam ir 20 ieeju/izeju pieslēgumu, no kurām daļu var izmantot dažādu specifisku uzdevumu veikšanai, piemēram, elektrodzinēju vadībai vai sensoru datu iegūšanai.

Mikrokontrollera komplektācijā ietverts viss nepieciešamais programmas izveidei un darbināšanai. Tas vienkārši jāpieslēdz pie jūsu datora USB (angļu val. — *Universal Serial Bus* — universālā seriālā kopne) pieslēgvietas, izmantojot atbilstošu kabeli (plašāk tas aplūkots zemāk). Mikrokontrolleri var izmantot kopā ar prototipēšanas plati vai arī ar robotu. Vienkāršākajos programmēšanas uzdevumos to var izmantot kā patstāvīgu iekārtu.

Ņemot vērā līdzību ar citiem Arduino tipa mikrokontrolleriem un jums ir pieredze to lietošanā, tas palīdzēs jums arī šajā grāmatā ietverto uzdevumu veikšanai.

Vispārīgas ziņas par mikrokontrolleri:

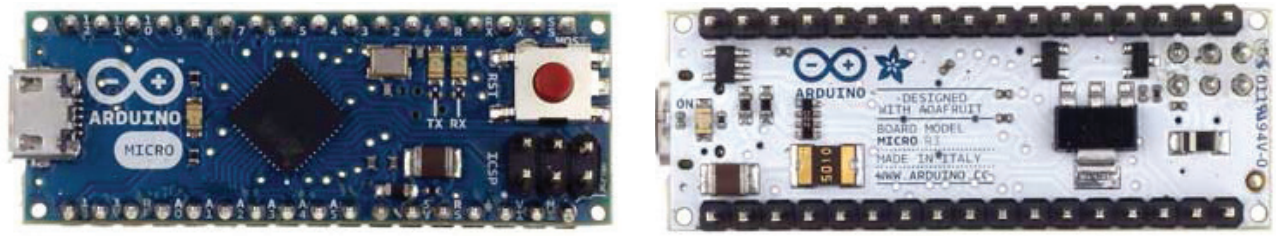
Parametrs	Vērtība
Izmantotais mikroprocesors	ATmega32u4
Darbības spriegums	5 V
Barošanas spriegums (ieteicamais)	7–12 V
Ciparu ieejas/izejas	20
PWM kanāli	7
Analogo signālu ieejas/izejas	12
Ieejas/izejas maksimālais strāvas stiprums	40 mA
Programmai pieejamā atmiņa	32 KB (kilobaiti), no tiem 4 KB aizņem sistēmas sāknēšanas programmatūra
Takts frekvence	16 MHz
Garums/platums	48 mm/18 mm
Svars	13 g

Detalizēts shēmu apraksts atrodams: http://arduino.cc/en/uploads/Main/arduino-micro-schematic.pdf.

Barošana

Lai nodrošinātu mikrokontrollera darbību, tas var tikt pieslēgts ārējam barošanas avotam vai USB pieslēgvietai. Mikrokontrolleris barošanas avotu nosaka automātiski. Ja tiek izmantoti ārēji barošanas avoti, kas nav USB, tad jāizmanto GND un VIN pieslēgvietas. Lai nodrošinātu mikrokontrollera normālu darbību, ražotājs iesaka izmantot 7–12 V spriegumu. Ja tas tiek pārsniegts, nesasniedzot 20 V, tad iespējama mikrokontrollera barošanas ķēžu pārkaršana. Ja barošanas spriegums ir zemāks par 7 V, tad iespējama nestabila mikrokontrollera darbība un darba rezultāts būs neprognozējams.

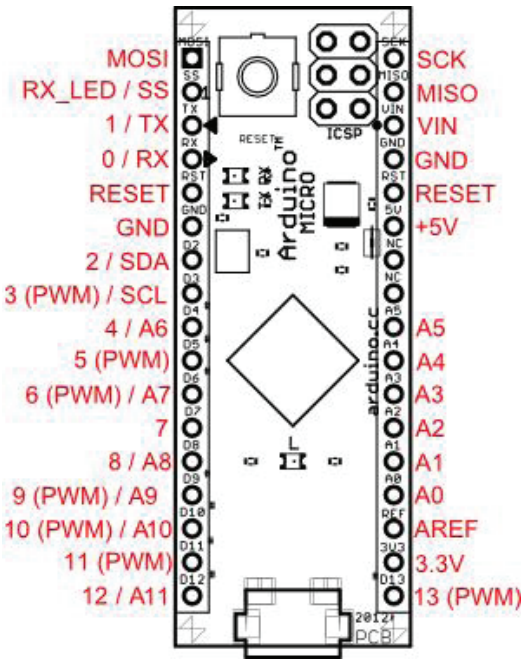
Mikrokontrollera ārējais izskats



Papildus minētajam mikrokontrolleris var nodrošināt nelielu ārējo ķēžu barošanu, pieslēdzot tās atbilstoši mikrokontrollera kāju (turpmāk — MK kāja) apzīmējumiem.

Apzīmējums	Paskaidrojums
VIN	Barošanas sprieguma ieeja, ja netiek izmantota USB pieslēgvieta, t. i., tiek izmantots ārējs barošanas avots.
5 V	5 V regulēts barošanas spriegums, ko var nodrošināt, gan izmantojot USB, gan arī VIN.
3 V	3,3 V barošanas spriegums ārēju ķēžu darbināšanai. Maksimālais strāvas stiprums, ko var nodrošināt šī izeja, ir 50 mA. Ja tas tiek pārsniegts, tad mikrokontrollera barošanas ķēdes var tikt neatgriezeniski bojātas.
GND	Zemes jeb 0 pieslēgvieta.

Pilna mikrokontrollera kāju shēma



Ieejas un izejas

Katra no mikrokontrollera 20 ieejām/izejām var tikt izmantota signālu izvadei vai iegūšanai, izmantojot komandas `pinMode()`, `digitalWrite()` un `digitalRead()`, kas sīkāk tiks aplūkotas programmēšanas pamatu apgūšanas nodaļās. Tās visas darbojas diapazonā no 0 līdz 5 V. Katra no ieejām/izejām spēj saņemt vai dot ne vairāk kā 40 mA lielu strāvu. Tām visām ir iekšēji slodzes rezistori diapazonā 20–50 kΩ.

Zemāk paskaidroti arī citi mikrokontrollera kāju apzīmējumi un to specifiskie pielietojumi.

Papildus minētajām ieejām/izejām mikrokontrolleris nodrošina arī citas specifiskas funkcijas, kas tiks aplūkotas vēlāk konkrētos uzdevumos.

Apzīmējums	Paskaidrojums
1/TX un 0/RX	Virknes komunikācijas savienojumam nepieciešamās ieejas/izejas. RX — datu saņemšanai, bet TX — datu nosūtīšanai ārējām iekārtām. Datu nosūtīšanai un saņemšanai signālu līmenis (spriegums) nedrīkst pārsniegt 5 V.
2/SDA un 3/SCL	Pieslēgvietas tiek izmantotas TWI datu apmaiņai (angļu val. — *Two Wire Interface* — divu vadu saskarne), kas ir alternatīvs virknes komunikācijas pārsūtīšanas un saņemšanas paņēmiens.
0 (RX), 1(TX),2,3	Papildus minētajām funkcijām šīs ieejas/izejas var tikt izmantotas ārējo pārtraukumu (angļu val. — *interrupts*) saņemšanai. Pārtraukumi tiek izmantoti, lai signalizētu galvenajai programmai par nepieciešamību pārtraukt tās darbu un izpildīt kādas īpašas darbības. Attiecīgo funkcionalitāti izmanto ar komandu `attachInterrupt()`.
PWM: 3,5,6,9,10,11 un 13	Šīs ieejas/izejas nodrošina 8 bitu impulsa platuma modulētu signālu uz pieslēgvietas, kas robotikā parasti tiek izmantota motoru vadībai, kā arī citiem specifiskiem pielietojumiem. Šo funkcionalitāti izmanto ar komandu `analogWrite()`.
LED 13	Papildu minētajai funkcionalitātei šī ieeja/izeja ir savienota ar sarkanas krāsas mirdzdiodi, kas pieejama vadībai no programmas.
A0–A5, A6–11	Mikrokontrollerim kopā ir 12 analogās ieejas/izejas ar 10 bitu izšķirtspēju. Tas nozīmē, ka sprieguma līmenis no 0 līdz 5 V tiek kodēts ar skaitli no 0 līdz 1023. Tomēr šos iestatījumus var mainīt ar komandas `analogReference()` palīdzību.

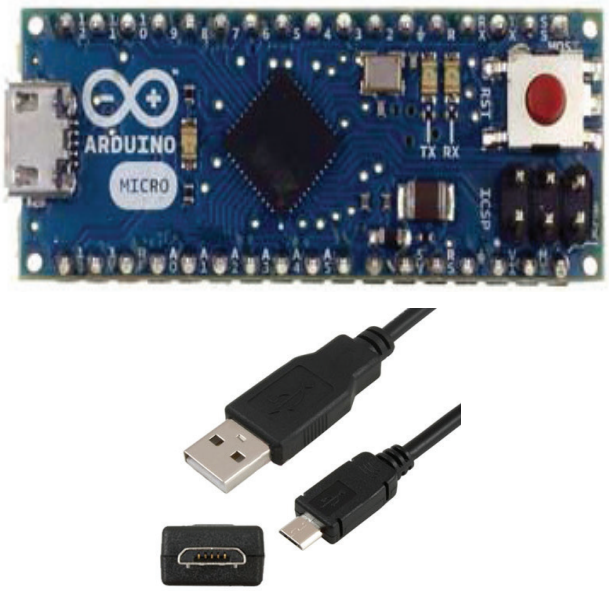
2.3. Programmēšanas vides instalēšana

Lai uzsāktu programmatūras izstrādi mikrokontrolleriem, jāinstalē un atbilstoši jākonfigurē izstrādes vide, kas sastāv no programmas redaktora un ArduinoMicro dziņa (angļu val. — *driver*). Zemāk norādīti visi nepieciešamie soļi, lai sagatavotu programmēšanas vidi operētājsistēmā *Windows 8*.

1. solis. Sagatavojiet Arduino Micro un USB kabeli

Lai instalētu izstrādes vidi, sagatavojiet Arduino Micro, kā arī USB mikrokabeli.

Nepieciešamie piederumi



USB mikrokabelis parasti ir daļa no moderno telefonu uzlādes piederumiem.

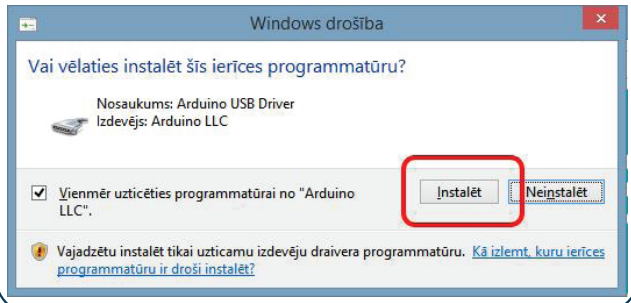
2. solis. Lejupielādējiet Arduino programmatūras izstrādes vidi

Vispirms lejupielādē vides instalēšanai nepieciešamos failus: http://arduino.cc/en/Main/Software. Izvēlieties operētājsistēmai un piekļuves tiesībām atbilstošus instalācijas failus.

Instalācijas failu izvēle Arduino lejupielādes mājaslapā

Ja lejupielādējāt instalētāju, palaidiet to, kad lejupielāde ir pabeigta. Ja lejupielādējāt ZIP arhīvu, tad atarhivējiet to un pēc tam palaidiet instalētāju. Sekojiet instalētāja norādēm. Ja operētājsistēma pieprasa atļaut instalēt iekārtas dzini, atļaujiet to.

Drošības jautājums par dziņa instalēšanu



3. solis. Arduino pieslēgšana

Pieslēdziet Arduino pie kādas no datora brīvajām USB pieslēgvietām. Zilas krāsas indikācijas mirdzdiode sāk nepārtraukti spīdēt. Tas nozīmē, ka Arduino mikrokontrolleris darbojas. Arī zaļā mirdzdiode mirgos, kas norāda uz ražotāja testa programmas darbību. Ja zaļā mirdzdiode nemirgo, tad tas nav jāuzskata par mikrokontrollera darbības traucējumu.

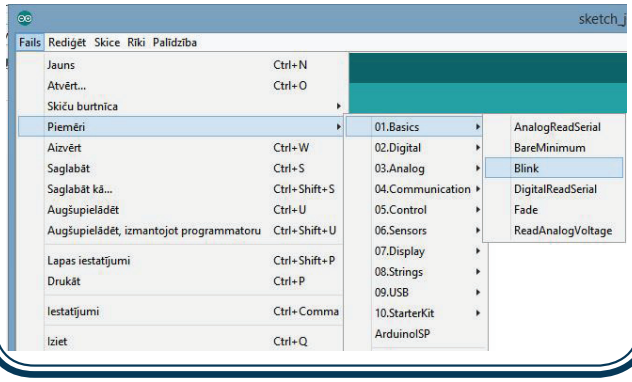
4. solis. Programmēšanas vides iedarbināšana

Ar dubultklikšķi uz programmēšanas vides darbvirsmas saīsnes iedarbiniet Arduino programmēšanas vidi. Vides valoda atbildīs jūsu operētājsistēmas iestatītajai, t. i., ja tā ir latviešu valoda, tad arī programmēšanas vides izvēlnes būs latviešu valodā. Ja tomēr tā nav, tad sekojiet norādēm, kas atrodamas: http://arduino.cc/en/Guide/Environment#languages.

5. solis. Atvērt piemēra programmu

Izvēlieties mirdzdiodes piemēra programmu.

Piemēra programmas atvēršana



Jaunā logā tiks attēlota piemēra programma, kuras būtība ir zaļās mirdzdiodes ieslēgšanas un izslēgšana ar intervālu viena sekunde.

Piemēra programma

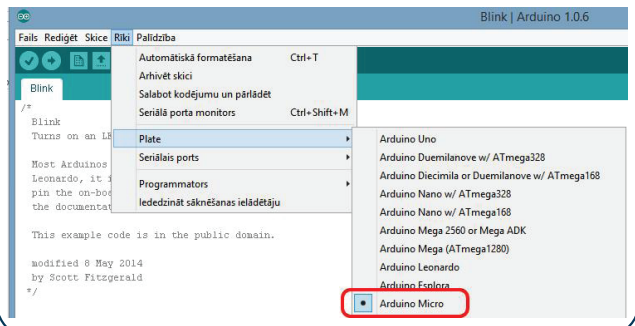
Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

6. solis. Mikrokontrollera izvēle

Izvēlieties izvēlni **Rīki > Plate > Arduino Micro** tā, kā tas norādīts attēlā.

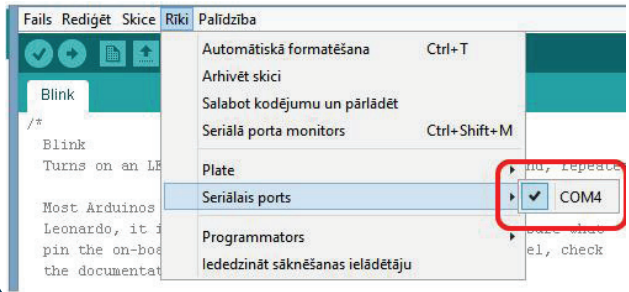
Mikrokontrollera veida iestatīšana



7. solis. COM porta iestatīšana

Lai nodrošinātu datu nosūtīšanu un saņemšanu uz mikrokontrolleri, nepieciešams iestatīt virknes komunikācijas pieslēgumu — COM portu. Tie tiek numurēti pēc kārtas, un Arduino mikrokontrolleriem parasti tas ir lielāks par COM3, t. i., COM4, COM5 utt (attēlā tas ir COM4).

Mikrokontrollera COM porta iestatīšana

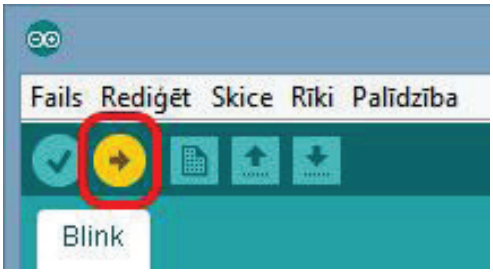


Ja neesat drošs par konkrētā porta numuru, atslēdziet mikrokontrolleri, pārbaudiet portu sarakstu vēlreiz. Īstais ir tas, kurš vairs nav atrodams sarakstā.

8. solis. Piemēra programmas nosūtīšana izpildei

Tagad atlicis nospiest pogu Augšupielādēt, dažas sekundes pagaidīt, kuru laikā var redzēt datu nosūtīšanas indikācijas — mirdzdiožu strauju mirgošanu (norāda uz datu nosūtīšanu vai saņemšanu) — un sagaidīt paziņojumu "Augšupielādēšana pabeigta".

Mikrokontrollera COM porta iestatīšana



Pēc dažām sekundēm zaļās krāsas mirdzdiode sāks mirgot ar vienas sekundes starplaiku. Ja tas tiešām tā ir, jūs esat izdarījis visu, lai sāktu apgūt programmēšanas pamatus!

Ja tomēr viss nenotiek tā, kā aprakstīts, tad, lūdzu, pārskatiet kādu no ieteikumiem, kas atrodami: http://arduino.cc/en/Guide/Troubleshooting.

Ja vēlaties patstāvīgi iepazīties ar mikrokontrollera iespējām vai programmēšanas pamatiem, aplūkojiet kādu no šiem informācijas avotiem:

- Piemēri dažādas sarežģītības uzdevumu veikšanai: http://arduino.cc/en/Tutorial/HomePage.
- Skaidrojumi par izmantoto programmēšanas valodu: http://arduino.cc/en/Reference/HomePage.

3. SĀKAM PROGRAMMĒT

3.1. Mērķis

Temata mērķis ir iepazīstināt ar programmēšanas iespējām, ko piedāvā konstruktorā izmantotais programmējamais mikrokontrolleris (turpmāk tekstā – MK) Arduino Micro. Šī temata uzdevums nav atklāt visas programmēšanas iespējas, bet koncentrēties konkrētu programmēšanas mērķu sasniegšanai. Tādēļ ir ieteicams aplūkot kādu no Arduino programmēšanas papildu mācību līdzekļiem, piemēram, B. Evansa "Arduino Programming Notebook", kas atrodama: http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf.

Šajā nodaļā tiek uzskatīts, ka esat veiksmīgi instalējis programmēšanas vidi, kā arī izpildījis 1. nodaļas "Elektronikas elementi" uzdevumus, uz kuriem šeit sastopamas atsauces.

3.2. Teorētiskā daļa

3.2.1. No kā sastāv MK Arduino programma

Katra Arduino programma sastāv no šādām daļām:

1. Globālo definīciju daļa – tajā tiek noteikti mainīgie, konstantes vai citas definīcijas, kas ir spēkā visā programmā, t. i., abās zemāk aprakstītajās programmas daļās.
2. Inicializācijas daļa – tiek izpildīta tikai vienreiz pirms pamata programmas izpildes. Šajā daļā parasti tiek definēti visi programmas gaitā nemainīgie lielumi, MK kāju nozīme (ieejas vai izejas), konstantes u. c. Būtiskākais, kas šajā daļā tiek noteikts, ir programmas laikā nepieciešamo ieeju vai izeju definēšana, kas nosaka, kā katra no ieejām/izejām varēs tikt izmantota. Inicializācijas daļas piemērs:

```
void setup()
//Funkcijas rezultāta datu tips un nosaukums
{   //Funkcijas sākums
    //Funkcijas ķermenis – visas izpildāmās
    //komandas
}   //Funkcijas beigas
```

Kā minēts, šī ir funkcija, kas tiks izpildīta vienreiz. Funkciju raksturo tās rezultāta datu tips (skaitlis, simbolu virkne vai kas cits.). Šajā piemērā atslēgas vārds void nozīmē, ka inicializācijas funkcijai nav rezultāta, t. i., tā tiek izpildīta vienu reizi un viss. Nosaukumam obligāti jābūt setup, lai MK iebūvētā programmas izpildes apakšsistēma spētu atšķirt programmas inicializāciju no citām programmas daļām.

3. Cikla daļa tiek izpildīta nepārtraukti, nolasa izejas, veic aprēķinus un izvada izejas signālus. Pēc pēdējās komandas izpildes programma atgriežas uz cikla daļas pirmo komandu un sāk izpildīt komandas no jauna. Šī ir galvenā programmas izpildes daļa, kur tiek ietverta visa programmas izpildes gaita un t.s. loģika. Cikla daļas piemērs:

```
void loop()
//Funkcijas rezultāta datu tips un nosaukums
{   //Funkcijas sākums
    //Cikla komandas
    //Funkcijas ķermenis – visas izpildāmās
    //komandas
}   //Funkcijas beigas
```

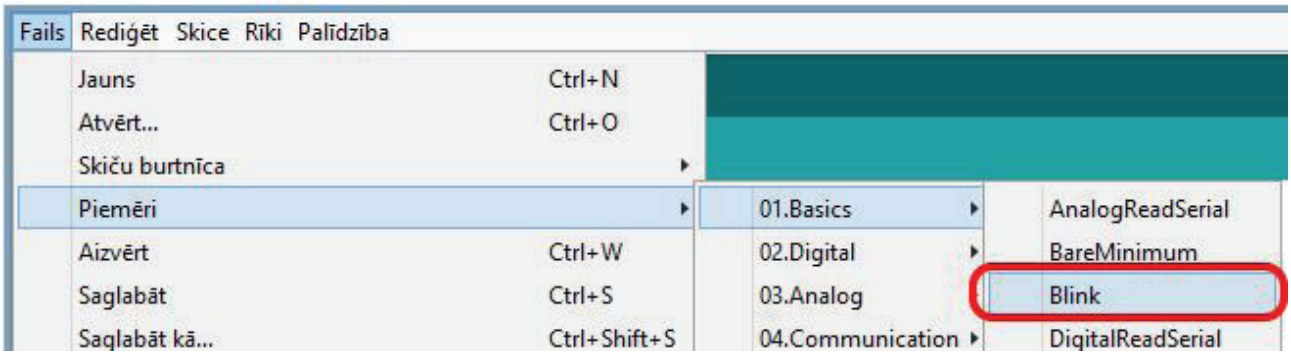
Arī šīs funkcijas rezultāta tips ir **void**, kas norāda, ka funkcijai nav rezultāta, t. i., tā tiek izpildīta ciklisko visas programmas darbības laikā.

Kā redzams, abu daļu struktūra ir vienāda, jo abas ir funkcijas, kuras atšķir tikai to nosaukumi *setup* un *loop*, un nozīme. Lai labāk izprastu katras funkcijas nozīmi, jāaplūko vienkāršs piemērs programmēšanas vides piemēru sadaļā **Fails → Piemēri → 01.Basics → Blink**.

Atveras šāds programmas kods:

```
void setup()
//Funkcijas rezultāta tips un nosaukums
{   //Funkcijas sākums
    pinMode(13,OUTPUT);
    //Definē 13. MK kāju kā izeju, t. i.,
    //to var izmantot signāla
}   //Funkcijas beigas
```

Programmu piemēru bāze



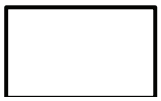
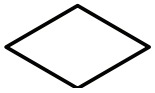

```
void loop()
    //Funkcijas rezultāta datu tips
    //un nosaukums
{   //Funkcijas sākums
  digitalWrite(13,HIGH);
    //13. izejā tiek uzstādīts izejas signāls
    //loģiskā 1. līmenī, t. i., +5 V
  delay(1000);
    //Programmas izpilde tiek apturēta
    //uz 1000 milisekundēm, t. i., uz vienu
    //sekundi
  digitalWrite(13,LOW);
    //13. izejā tiek uzstādīts izejas signāls
    //loģiskā 0 līmenī, t. i., 0 V
  delay(1000);
    //Atkal izpilde tiek apturēta
    //uz 1000 milisekundēm, t. i.,
    //uz vienu sekundi
}
```

Ko var redzēt šajā programmas kodā? Tātad:

1. Ir noteikts, ka 13. MK kāja ir izeja. Ar tās palīdzību var izvadīt kādu signālu. Šajā piemēra programmā tai periodiski tiek noteikts izejas signāls loģiskā 1 (+5 V) un 0 (0 V) līmenī. Ņemot vērā, ka 13. izeja vienlaicīgi ir savienota ar MK Arduino esošu mirdzdiodi, šādi, mainot izejas signāla vērtību, panāk, ka mirdzdiode tiek periodiski ieslēgta vai izslēgta.
2. Ir izveidota nepārtraukti izpildāmā funkcija **loop()**, kas katram signāla līmenim atvēl vienu sekundi jeb 1000 milisekundes. Tas tiek panākts, apturot programmas izpildi. Kamēr programma nemaina ieejas/izeju stāvokli, tajos nekas arī nemainās. Tādējādi, izvadot signāla līmeni +5 V un pēc tam apturot programmas izpildi, mirdzdiode turpinās spīdēt, līdz izejas signāla līmenis netiks samazināts līdz 0 V līmenim.
3. Pēdējā rindiņa nosaka, ka programma arī ar 0 V līmeni tiek apturēta uz vienu sekundi. Tādējādi mirdzdiodes ieslēgšanas un izslēgšanas periodi ir vienādi. Pēc šīs komandas programma atgriežas uz **loop()** funkcijas pirmo komandu, un tās izpilde sākas no sākuma.

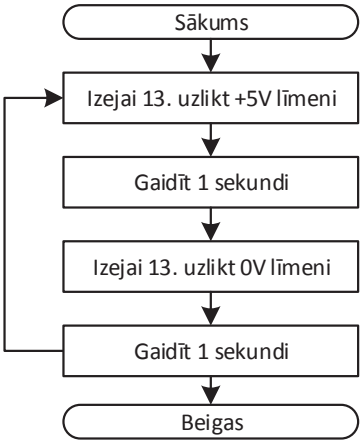
3.2.1.1. Programmas attēlojums ar diagrammu

Dažkārt sarežģītus algoritmus ir ērti attēlot grafiskas diagrammas veidā. Viens no visizplatītākajiem šādu diagrammu veidiem ir bloku diagrammas jeb blokshēmas. To pamatelementi izskatās šādi.

	Programmas sākums vai beigas.
	Programmas solis vai kāda cita izpildāma darbība.
	Pārbaude, nosacījums tālākai programmas izpildei.
	Programmas izpildes gaita.

Izmantojot šos blokshēmu pamatelementus, piemēra funkciju **loop()** iespējams attēlot ar diagrammas palīdzību, kas redzama attēlā. Turpmākajos uzdevumos izveidotās funkcijas tiks attēlotas arī ar blokshēmas palīdzību, kas ļaus ērtāk apgūt un attēlot arī sarežģītākas programmas izpildes gaitas.

Piemēra programmas izpildes blokshēma



3.2.1.2. Programmas pamatkonstrukcijas

Šajā apakšnodaļā īsi tiks aplūkotas MK Arduino programmas galvenās konstrukcijas.

Funkcijas

Funkcija ir komandu kopums, kas tiek izpildīts vienmēr, kad funkcija tiek izsaukta. Divas funkcijas jau ir zināmas — **setup()** un **loop().** Programmētājs parasti savas programmas būtību cenšas ietvert vienā vai vairākās sevis izveidotās funkcijās, kuras tiek izsauktas no **setup()** vai **loop()**. Funkcijai ir šāda struktūra:

```
tips funkcijasNosaukums(argumenti)
{
   komandas;
}
```

Varam izveidot savu piemēra funkciju, kas ļauj periodiski izslēgt un ieslēgt mirdzdiodi (atbilstoši jau aplūkotajam piemēram):

```
void piemeraFunkcija()
{
   digitalWrite(13,HIGH);
   delay(1000);
   digitalWrite(13,LOW);
   delay(1000);
}
```

Lai to izsauktu, nedaudz jāmodificē loop() funkcija:

```
void loop()
{
   piemeraFunkcija();
}
```

Programmas kods Arduino vidē izskatās šādi:

```
void loop()
{
   piemeraFunkcija();
}

void piemeraFunkcija()
{
   digitalWrite(13,HIGH);
   delay(1000);
   digitalWrite(13,LOW);
   delay(1000);
}
```

Ja vēlamies iegūt kādu konkrētu rezultātu no funkcijas, tad jānorāda funkcijas atgriežamais tips:

```
int divuSkaitluSumma(int x, int y)
{
   return (x+y);
}
```

Mainīgie

Mainīgie ir veids, kā īslaicīgi saglabāt un nosaukt (dot vārdu) noteiktiem skaitliskiem vai cita veida datiem. Pats nosaukums pasaka priekšā — mainīgo vērtības var mainīt programmas izpildes laikā, kad vien tas nepieciešams. Katram mainīgajam ir tā datu tips (skat. zemāk), nosaukums un vērtība. Zemāk redzamajā piemērā no jauna deklarētam (definētam) mainīgajam tiek piešķirta sākuma vērtība un pēc tam tā tiek mainīta:

```
int mansPiemeraMainigais = 0;
   //Mainīgā definīcija un sākuma vērtības
   //piešķiršana
mansPiemeraMainigais = 32;
   //Mainīgā vērtības maiņa
```

Uzmanību! *Mainīgo nosaukumiem ir jābūt jēgpilniem, lai programmas kods būtu saprotams arī citiem.*

Ir jāatceras, ka mainīgais ir līdzīgs mantu glabātuvei, respektīvi, tas atspoguļo MK atmiņas saturu. Saturs ir pieejams vienmēr, un tas nemainās, iekams tas netiek mainīts. Nākamajā piemērā tiek izmantota iepriekš piešķirtā vērtība, lai to palielinātu par 10.

```
mansPiemeraMainigais = mansPiemeraMainigais + 10;
   //Mainīgā vērtībai tiek pieskaitīts 10
```

Uzmanību! Atcerieties, ka '=' nav vienādības zīme, bet gan vērtības piešķiršana. Iepriekšējā piemēra komanda nav salīdzināšana kā matemātikā, bet gan esošās vērtības izmaiņa. Darbības būtība ir šāda: iegūt mainīgā *mansPiemeraMainigais* vērtību, kas šajā gadījumā ir 32, un pieskaitīt tai 10. Rezultātu noglabāt mainīgajā *mansPiemeraMainigais*. Tātad mainīgā *mansPiemeraMainigais* vērtība būs 42.

Mainīgā vērtības bieži jāpārbauda, lai pieņemtu lēmumu par programmas tālāku izpildi. Zemāk redzamajā piemērā tiek veikta analogās vērtības nolase no 2. MK kājas, pēc tam tās vērtība tiek salīdzināta. Ja vērtība ir lielāka par 100, tad mainīgajam tiek piešķirta vērtība 100 (piemērs no: http://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf):

```
int ieejasMainigais = 0
   //Definē mainīgo un tā sākuma vērtību
ieejasMainigais = analogRead(2);
   //Nolasa analogā signāla vērtību
   //no 2. MK kājas un ieraksta vērtību
   //mainīgajā}
if (ieejasMainigais > 100)
   //Tiek veikta mainīgā vērtības
   //salīdzināšana ar 100
{
ieejasMainigais = 100;
   //Ja vērtība lielāka par 100, tad piešķir
   //vērtību 100, t. i., mainīgā vērtība tiek
   //ierobežota ar augšējo slieksni 100
}
delay(ieejasMainigais);
```

Mainīgo darbības zona

Katram mainīgajam ir tā darbības zona — redzamība, kurā tā vērtība ir pieejama. Mainīgo var definēt programmas pašā sākumā — globālie mainīgie, funkcijas ķermenī vai dažkārt pat komandu blokā, piemēram, ciklā. Mainīgā definēšanas vieta nosaka tā redzamību. Nākamajā piemērā doti dažādi mainīgie, kuru redzamība ir atšķirīga:

```
int globalaisMainigais;
  //Šis mainīgais ir redzams visā programmā
void setup()
{
    int setupMainigais;
      //Šis mainīgais ir pieejams tikai setup()
}
void loop()
{
    int loopMainigais;
      //Šis mainīgais ir pieejams tikai loop()
      //funkcijas ķermenī
    for(int i = 0; i < 20; i = i + 1)
      //Šajā ciklā tiek definēts jauns
      //mainīgais i, kas ir pieejams tikai šī
      //cikla ķermenī. Tādēļ šādus mainīgos
      //dažkārt sauc par cikla mainīgajiem
   {  //Citas komandas
   }
}
```

Datu tipi

Katram mainīgajam var būt savs datu tips, kas nosaka tā vietu MK atmiņā, kā arī tā izmantošanas veidu. Neraugoties uz datu tipu daudzveidību, šeit tiek aplūkoti tikai galvenie.

Datu tips	Skaidrojums
1.1.1.1.1.1 byte	8 bitu skaitlisks tips, kas spēj uzglabāt skaitļus no 0 līdz 255. *byte* `piemeraMainigais = 123;`
1.1.1.1.1.2 int	16 bitu vesels skaitlis, kas var saturēt vērtības no −32 767 līdz 32 768 *int* `piemeraMainigais = 12 300;`
1.1.1.1.1.3 float	Reālo skaitļu datu tips, kas aizņem 32 bitus un spēj uzglābāt skaitļus no apmēram $3,4 \cdot 10^{38}$ līdz $-3,4 \cdot 10^{38}$. *float* `piemeraMainigais = 12300.546;`

Masīvi	Masīvi ir vienādu datu tipu kopa, kurai iespējams piekļūt ar kārtas numuru vai indeksu. Pirmā elementa indekss vienmēr ir 0. Masīvam sākumā var piešķirt visas vērtības vai arī darīt to pakāpeniski programmas izpildes gaitā.
	int `piemeraMasivs[] = {12,-3,8,15};`
	Te tiek izveidots masīvs ar nosaukumu **piemeraMasivs** un datu tipu **int**. Tam tiek piešķirtas sākotnējās vērtības. Tas nozīmē, ka masīva 0. elementa vērtība ir 12, bet 3. elementa vērtība ir 15.
	int `citsMainigais = piemeraMasivs[1];`
	Pēc šīs komandas izpildes mainīgā citsMaingais vērtība ir −3.
	Masīvi ir ērti apstrādei ciklu ietvaros. Nākamais piemērs rāda, kā iepriekš definētajā masīvā automātiski saglabā nepieciešamās vērtības no analogo signālu ieejas.
	`for(int i = 0; i < 4; i = i +1)`
	`{`
	`  piemeraMasivs[i] = analogRead(2);`
	`}`
	Piemēra ciklu sāk ar 0. indeksu (i = 0), kas tiek palielināts par 1, kamēr vien tas ir mazāks par 4. Tas nozīmē, ka pēdējo reizi cikls tiks izpildīts tad, kad i vērtība būs 3, jo, ja i vienāds ar 4, tad nevienādība i < 4 nebūs patiesa un cikla darbība tiks pārtraukta.

Bool	Šī tipa mainīgie var pieņemt vērtību *patiess* vai *aplams*. Atbilstoši šim nosacījumam Arduino programmēšanas vide ļauj piešķirt šiem mainīgajiem šādas vērtības: Patiess — TRUE Aplams — FALSE Loģiskais 1 (+5 V) — HIGH Loģiskais 0 (0 V) — LOW

Loģiskie operatori un salīdzināšana

Salīdzināšana jau tika demonstrēta piemēra programmās ar operatora **if** starpniecību. Tomēr ir būtiski redzēt visu salīdzināšanas operatoru daudzveidību, lai nodrošinātu ātru un ērtu programmas koda izveidi:

```
a==b   //mainīgais a ir vienāds ar mainīgo b

a!=b   //mainīgais a nav vienāds ar mainīgo b

a<b    //a mazāks par b

a>b    //a lielāks par b

a<=b   //a mazāks vai vienāds par b

a>=b   //a lielāks vai vienāds ar b
```

Visu loģisko izteiksmju rezultāts ir *patiess* vai *aplams*. Šādi rakstītas loģiskās izteiksmes var kombinēt ar loģiskajiem operatoriem, lai veidotu sarežģītus nosacījumus.

```
if(a>b && a<45)
  // && operators tulkojams kā 'un', t. i.,
  //lai visas izteiksmes vērtība būtu patiess,
  //izteiksmei a>b, kā arī a<45 ir jābūt
  //patiesām.

if(a>b || a<45)
  // || operators tulkojams kā 'vai', t. i.,
  //lai visas izteiksmes vērtība būtu patiess,
  //vienai no izteiksmēm a>b un a<45 ir jābūt
  //patiesai.

if(!a>b)
  // ! operators tulkojams kā 'ne', t. i.,
  //lai visas izteiksmes vērtība būtu patiess,
  //izteiksmei a>b ir jābūt aplamai.
```

Nosacījuma operators

Nosacījuma operators ļauj pieņemt lēmumu par programmas tālāku izpildi. Piemēram, ja nepieciešams pieņemt lēmumu par konkrēta ieejas signāla apstrādi, var izmantot šādu konstrukciju:

```
int ieejasMainigais = analogRead(2);

if(ieejasMainigais > 100)
{
   //Veikt signāla apstrādes vai citas darbības
}
else
{
   //Veikt alternatīvu darbību secību
}
```

Kā redzams, jau zināmajam operatoram **if** tiek pievienota alternatīvas sekcija **else**, kas ļauj veikt kādas citas darbības, ja ieejas signāla vērtība tomēr nav lielāka par 100, kā norādīts piemērā. Viens nosacījuma operators var ietvert citus, tādējādi veidojot sarežģītus programmas izpildes scenārijus, piemēram:

```
int ieejasMainigais = analogRead(2);
if(ieejasMainigais > 100)
{
   //Veikt signāla apstrādes vai citas darbības
   if(ieejasMainigais < 130)
   {
      //Īpaša ieejas signāla vērtības
      //apstrāde, kad tā ir starp 100 un 130
   }
}
else if (ieejasMainigais < 30)
{
   //Veikt signāla apstrādes vai citas
   //darbības, ja ieejas signāla vērtība
   //ir mazāka par 30
}
else
{
   //Veikt alternatīvu darbību secību,
   //ja neviens no nosacījumiem nedarbojas
}
```

FOR cikla operators

Kā tika demonstrēts iepriekš, cikla operators ļauj noteiktu reižu skaitu izpildīt vienas un tās pašas darbības. Tādējādi cikli ir līdzīgi **loop()** funkcijai, bet ļauj kontrolēt cikla izpildi. Katru reizi, kad tiek izpildītas visas cikla ķermenī esošās komandas, tiek veikta viena cikla *iterācija*. Tādējādi cikls ir viena no fundamentālām programmēšanas tehnikām, kas ir pamats visām programmām un automatizācijai kopumā. FOR cikla konstrukcija ir šāda:

```
for (inicializācija ; nosacījums ;
     darbība ar cikla mainīgo)
{
   //Cikla ķermeņa darbības
}
```

- Inicializācijas sekcijā parasti tiek inicializēta cikla mainīgā vērtība, kas var būt vai nebūt vienāda ar 0.
- Nosacījums ļauj vadīt cikla iterāciju skaitu, jo nosaka, kā kārtējā cikla iterācija tiek izpildīta.
- Darbības ar cikla mainīgo ļauj noteikt cikla iterāciju skaitu.

Tipiska FOR cikla piemērs redzams zemāk:

```
for (int i = 0; i < 4; i = i + 1)
{
   digitalWrite(13,HIGH);
   delay(1000);
   digitalWrite(13,LOW);
   delay(1000);
}
```

Šāda cikla izpildes rezultātā uz MK esošā mirdzdiode tiks ieslēgta un izslēgta četras reizes.

While cikla operators

Šis cikla operators ir ļoti līdzīgs jau apskatītajam FOR, tomēr tas nesatur cikla mainīgo un tādēļ ļauj izpildīt iepriekš nezināmu skaitu iterāciju. Cikla vadība tiek realizēta ar nosacījumu, kas tam jābūt patiesam, lai tiktu izpildīta kārtējā iterācija. WHILE cikla konstrukcija ir šāda:

```
while (nosacījums, kas ir patiess)
{
   //Cikla ķermenis
}
```

Tādējādi WHILE cikls var kalpot par labu instrumentu iepriekš neprognozējamas programmas izpildei. Piemēram, ja nepieciešams nogaidīt, iekams ieejas signāls nonāk noteiktā sprieguma līmenī, var izmantot šādu konstrukciju:

```
int ieejasMainigais = 0;
while (ieejasMainigais < 100)
{
   digitalWrite(13,HIGH);
   delay(10);
   digitalWrite(13,LOW);
   delay(10);
   ieejasMainigais = analogRead(2);
}
```

Cikls nodrošinās mirdzdiodes mirgošanu, kamēr ieejas signāls nonāks noteiktā līmenī. Tādējādi lietotājam tiek signalizēts, ka ieejas signāla līmenis ir jāpaaugstina.

3.1. DARBA LAPA. Poga un mirdzdiode

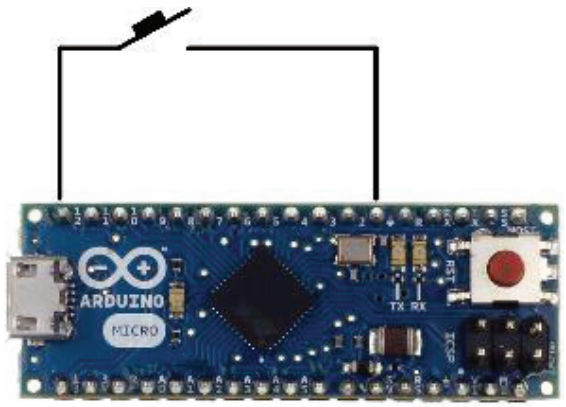
Mērķis

Izveidot savu pirmo programmu, kas, reaģējot uz lietotāja nospiestu pogu, ieslēdz mirdzdiodi.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Montāžas vads	1 gab.

Izveidojamā shēma



Šāda shēma ļauj izveidot ķēdi, kas nodrošina mirdzdiodes spīdēšanu tad, kad tiek nospiesta poga. Reakciju uz pogas spiedienu atšķirībā no elektronikas pamatu nodaļā apskatītajiem piemēriem nodrošinās MK Arduino. Konstruktorā pogas **P1** un **P2** ir pieslēgtas pie kopējā izvada — 0 pola. Tas nozīmē, ka reālo shēmu var vienkāršot tā, kā tas norādīts attēlā "Nepieciešamie materiāli un slēgums ", t. i., nav jāizmanto papildu detaļas.

Darba izpildes soļi

1. Izveidojiet slēguma shēmu.
2. Sastādiet doto programmu.
3. Ierakstiet programmu MK Arduino atmiņā tā, kā tas tika norādīts programmas izstrādes vides sagatavošanas nodaļā.
4. Ar baudu nospiediet pogu. ☺

```
void setup()
{
   pinMode(13,OUTPUT);
      //Nosakam, ka 13. MK kāja ir izeja
   pinMode(12,INPUT);
      //Nosakam, ka 12. MK kāja ir ieeja,
      //kas nolasīs pogas stāvokli
   digitalWrite(12,HIGH);
      //Pieslēdzam iekšējo rezistoru,
      //lai nebūtu jālieto papildu shēmas
}
void loop()
{
   bool PogasStavoklis = digitalRead(12);
      //Nolasām 12. MK kājas stāvokli
   digitalWrite(13,not(PogasStavoklis));
   //Iestatām 13. MK kājas stāvokli
   //atbilstoši nolasītajai vērtībai
}
```

Īss skaidrojums

Inicializācijas daļa:

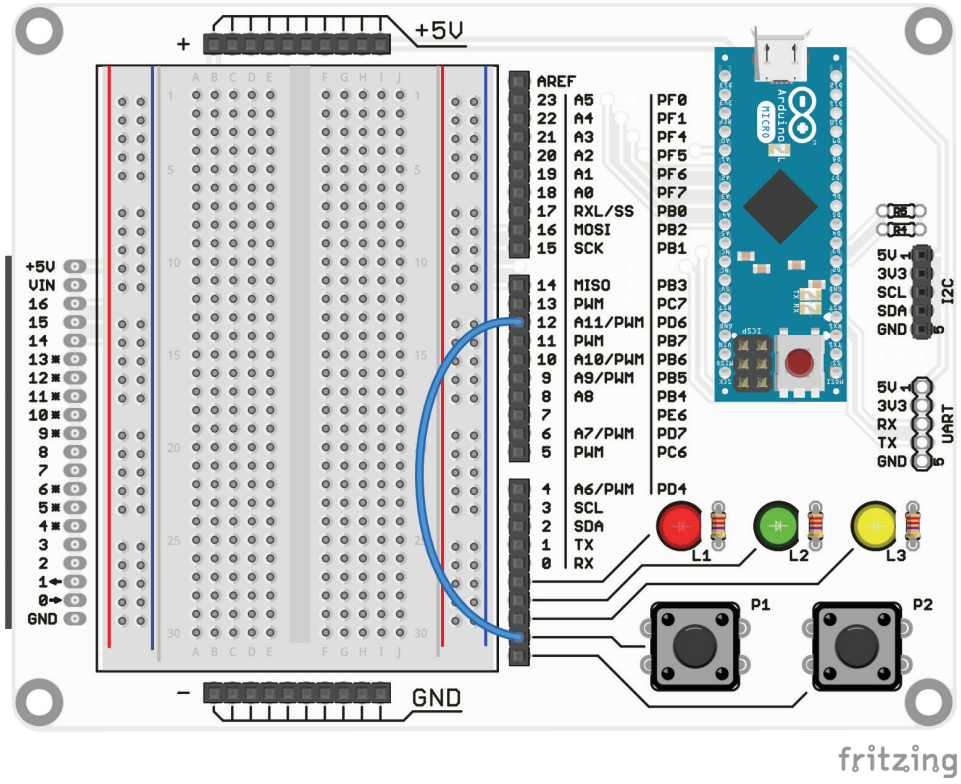
- Līdzīgi kā iepriekš, 13. MK kāja jānosaka par mirdzdiodes izeju;
- 12. MK kāju nosaka par ieeju, kas nepieciešama pogas stāvokļa nolasīšanai;
- 12. MK kājai tiek pieslēgts rezistors, lai nevajadzētu veidot ārējās ķēdes. Šis rezistors nodrošina 12. MK kājas loģiskā 1 līmenī, t. i., +5 V līmeni vienmēr, kamēr nav citi signāla avoti.

Cikla daļa:

- Tiek izveidots stāvokļa mainīgais **PogasStavoklis** ar loģisko tipu **bool**. Tātad tas var pieņemt divas vērtības *patiess* (loģiskais 1 vai +5 V) vai *aplams* (loģiskā 0 vai 0 V). Mainīgie tiek izmantoti īslaicīgai informācijas uzglabāšanai. Kamēr to vērtība netiek mainīta, tā arī nemainās neatkarīgi no programmas izpildes gaitas. Šajā piemērā, ja viena cikla izpildes laikā mainīgajam ir piešķirta noteikta vērtība, tad tā nemainās nākamajās cikla izpildes reizēs, iekams to nenomaina kāds notikums vai komanda programmā.

- Komanda **digitalRead(12)** ļauj nolasīt 12. MK kājas loģisko vērtību (0 vai 1). Nolasītā vērtība tiek ierakstīta mainīgajā **PogasStavoklis**.
- Komanda **digitalWrite(13,....)** ļauj instalēt norādītās MK kājas (13) līmeni.
- Šajā programmā nepieciešams izmantot vēl papildu operatoru **not(),** kas iekavās norādīto loģisko vērtību pārvērš uz pretējo, t. i., ja tiek norādīts loģiskā 0, tad tā tiek pārvērsta par loģisko 1 un otrādi. Te tas nepieciešams tādēļ, ka tiek izmantots iekšējais rezistors, t. i., ja nav citu signālu, tad 12. MK kājas līmenis būs +5 V jeb loģiskais 1. Ja netiks izmantots operators **not()**, mirdzdiode tiks ieslēgta tad, kad poga nav nospiesta, t. i., pretēji ierastajam.

Shēma



3.2. DARBA LAPA. Kāpņutelpas apgaismojums

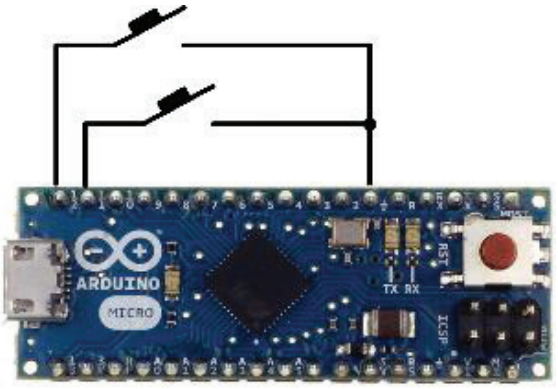
Mērķis

Izveidot programmu, kas ļauj ar divām pogām ieslēgt vai izslēgt mirdzdiodi, imitējot kāpņutelpu, t. i., ieslēdzot ar vienu pogu, var izslēgt ar citu.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Montāžas vadi	2 gab.

Izveidojamā shēma



Šāda shēma ļauj izveidot ķēdi, kas nodrošina mirdzdiodes spīdēšanu tad, kad tiek nospiesta poga. Reakciju uz pogas spiedienu atšķirībā no elektronikas pamatu nodaļā apskatītajiem piemēriem nodrošinās MK Arduino. Konstruktorā pogas **P1** un **P2** ir pieslēgtas pie kopējā izvada — 0 pola. Tas nozīmē, ka reālo shēmu var vienkāršot tā, kā tas norādīts attēlā "Nepieciešamie materiāli un slēgums ", t. i., nav jāizmanto papildu detaļas.

Darba izpildes soļi

1. Izveidojiet slēguma shēmu.
2. Sastādiet doto programmu.
3. Ierakstiet programmu MK Arduino atmiņā tā, kā tas tika norādīts programmas izstrādes vides sagatavošanas nodaļā.
4. Izbaudiet paša izveidoto kāpņutelpas apgaismojuma sistēmu.

```
bool MirdzdiodesStavoklis = false;
   //Mainīgajā tiks uzglabāta mirdzdiodes
   //vērtība
void setup()
{
   pinMode(13,OUTPUT);
      //Nosakam, ka 13. MK kāja ir izeja
   pinMode(12,INPUT);
      //Nosakam, ka 12. MK kāja ir ieeja,
      //kas nolasīs pogas stāvokli
   digitalWrite(12,HIGH);
      //Pieslēdzam iekšējo rezistoru,
      //la nebūtu jālieto papildu shēmas
   pinMode(11,INPUT);
      //Nosakam, ka 12. MK kāja ir ieeja,
      //kas nolasīs pogas stāvokli
   digitalWrite(11,HIGH);
      //Pieslēdzam iekšējo rezistoru,
      //la nebūtu jālieto papildu shēmas
}
void loop()
{
   bool PogasStavoklis_1 = digitalRead(12);
      //Nolasām 12. MK kājas stāvokli
   bool PogasStavoklis_2 = digitalRead(11);
      //Nolasām 11. MK kājas stāvokli
   if((PogasStavoklis_1 && PogasStavoklis_2)
                                  == false)
     //Pārbaude, vai kāda no pogām ir nospiesta
   {
      MirdzdiodesStavoklis =
                  not(MirdzdiodesStavoklis);
        //Ja ir, tad mirdzdiodes stāvoklis tiek
                        mainīts uz pretējo
   }
   digitalWrite(13,MirdzdiodesStavoklis);
      //Iestata jauno mirdzdiodes stāvokli
delay(300);
}
```

Īss skaidrojums

Globālie mainīgie:

- Ir pievienots viens mainīgais, kas ir pieejams abās funkcijās — **setup()** un **loop()**, kas paredzēts mirdzdiodes stāvokļa saglabāšanai ārpus funkcijas kārtējās izpildes cikla.

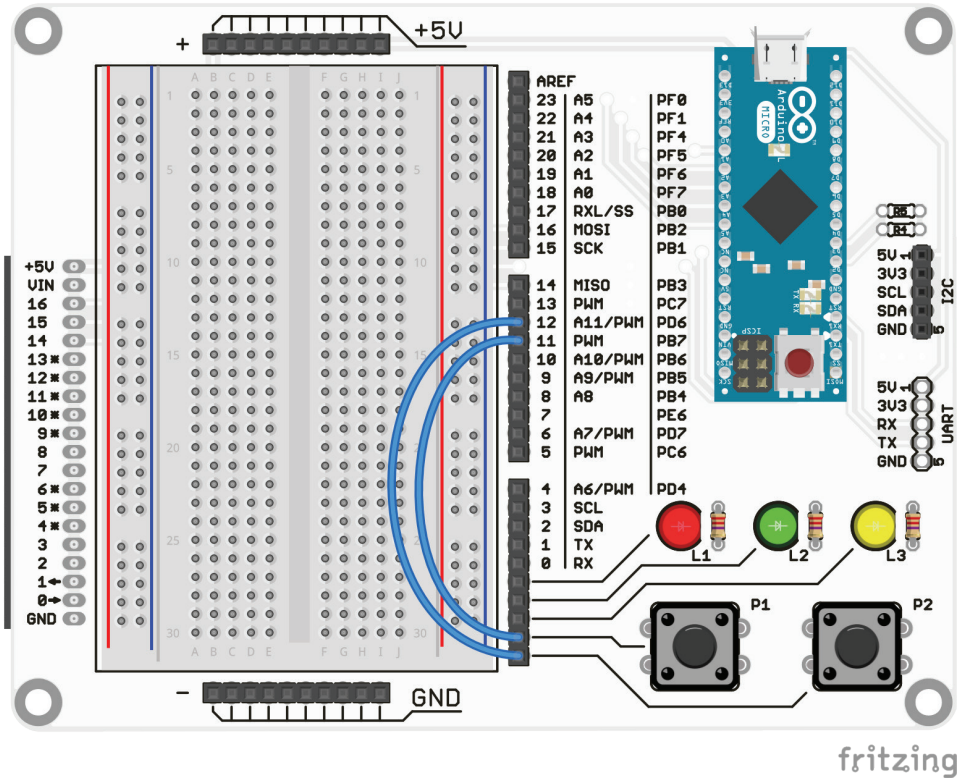
Inicializācijas daļa:

- Iepriekšējā uzdevuma inicializācijas daļa papildināt ar vēl vienu pogas ieeju — 11. MK kāju, kas definēta precīzi kā 12. MK kāja.

Cikla daļa:

- Līdzīgi kā iepriekšējā uzdevumā, ir definēti divu pogu stāvokļi **Pogas_Stavoklis_1** un **Pogas_Stavoklis_2**, kas saturēs atbilstoši 11. un 12. MK kājas stāvokli viena cikla ietvaros.
- Ir ieviests nosacījums **if** — nosacījums, kas ļauj novērtēt pogu stāvokļus. Šeit tiek pārbaudīts, vai kāda no pogām ir nospiesta. Šim nolūkam tiek izmantots loģiskais operators && — loģiskais UN. Šajā gadījumā tas dos vērtību patiess (loģiskais 1) tad un tikai tad, ja abi operandi — pogu stāvokļi — ir loģiskā 1 līmenī, t. i., neviena no tām nav nospiesta un iekšējais rezistors nodrošina loģiskā 1 sprieguma līmeni.
- Ja kāda no pogām nav nospiesta, tad nosacījuma vērtība ir vienāda ar loģisko 0 jeb **false** (aplams). Šajā gadījumā esošais mirdzdiodes stāvoklis ir jāmaina uz pretējo. Ja mirdzdiode mirdz, tad tā ir jāizslēdz, bet, ja nemirdz, tad jāieslēdz.
- Visbeidzot tiek iestatīts jaunais mirdzdiodes stāvoklis.
- Lai programmas darbības ātrums atbilstu cilvēka rīcības ātrumam, ir jāievieš neliela programmas izpildes pauze, ko nodrošina komanda `delay(300)` — aptur izpildi uz 300 milisekundēm.

Shēma



3.3. DARBA LAPA. Mirdzdiodes spožums

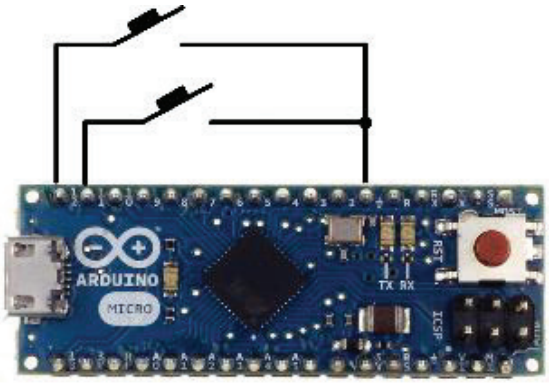
Mērķis

Izveidot programmu, kas ļauj ar divām pogām regulēt mirdzdiodes spožuma izmaiņas — ar vienu pogu spožums pieaug, bet ar otru samazinās, līdz pilnīgi izdziest.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Montāžas vads	2 gab.

Izveidojamā shēma



Šāda shēma ļauj izveidot ķēdi, kas nodrošina mirdzdiodes spožuma regulēšanu. Kā vienmēr, pats svarīgākais ir programmas izpildes loģika. Konstruktorā pogas **P1** un **P2** ir pieslēgtas pie kopējā izvada — 0 pola. Tas nozīmē, ka reālo shēmu var vienkāršot tā, kā tas norādīts attēlā "Nepieciešamie materiāli un slēgums ", t. i., nav jāizmanto papildu detaļas.

Darba izpildes soļi

1. Izveidojiet slēguma shēmu.
2. Sastādiet doto programmu.
3. Ierakstiet programmu MK Arduino atmiņā tā, kā tas tika norādīts programmas izstrādes vides sagatavošanas nodaļā.
4. Izbaudiet paša izveidotā spožuma mainītāja darbību.

```
int ledKaja = 13;
int pieaugums = 0;
int spozums = 127;

void setup()
{
   pinMode(ledKaja,OUTPUT);
   analogWrite(ledKaja,spozums);
   pinMode(11,INPUT);
   pinMode(12,INPUT);
}
void loop()
{
   bool PogasStavoklis_1 = digitalRead(12);
    //Nolasām 12. MK kājas stāvokli
   bool PogasStavoklis_2 = digitalRead(11);
      //Nolasām 11. MK kājas stāvokli
   if(PogasStavoklis_1 == true)
                              { pieaugums = 5; }
   if(PogasStavoklis_2 == true)
                             { pieaugums = -5; }
   spozums = spozums + pieaugums;
   if(spozums >= 255) { spozums = 255; }
   if(spozums <= 0) { spozums = 0; }
   analogWrite(ledKaja,spozums);
   delay(30);
}
```

Īss skaidrojums

Globālie mainīgie:

- Šeit tiek definēti trīs mainīgie:
  - **ledKaja** — norāda uz MK iebūvētu mirdzdiodi.
  - **pieaugums** — spožuma pieaugums. Ja tas ir pozitīvs, mirdzdiodes spožums lēnām sasniedz maksimālo, bet, ja negatīvs, tad lēnām nodziest.
  - **spožums**, kas tiek iestatīts mirdzdiodes spožuma noteikšanai. Šis mainīgais var ieņemt vērtības no 0 līdz 255.

Inicializācijas daļa:

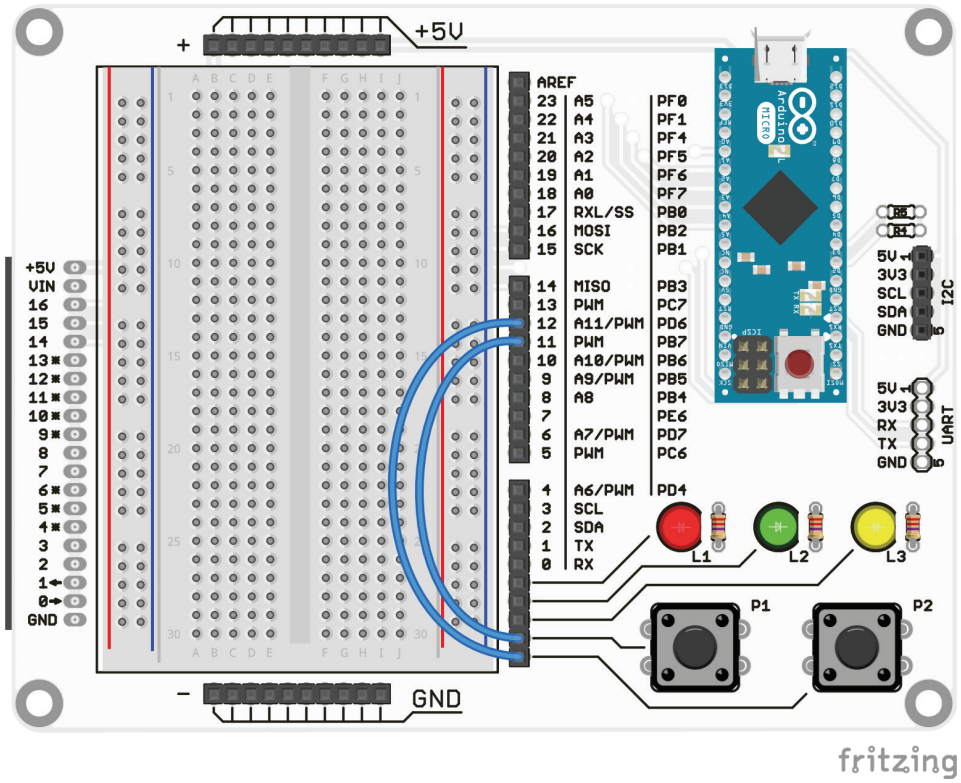
- Līdzīgi kā līdz šim — tiek noteiktas ieejas un izejas, bet papildus tiek noteikts arī sākotnējais mirdzdiodes spožums.

Cikla daļa:

- Līdzīgi kā iepriekšējā uzdevumā, ir definēti divu pogu stāvokļi **Pogas_Stavoklis_1** un **Pogas_Stavoklis_2**, kas saturēs atbilstoši 11. un 12. MK kājas stāvokli viena cikla ietvaros.
- Ir ieviesti nosacījumi, kas ļauj mainīt mirdzdiodes spožuma pieaugumu attiecīgi 5 vai −5, atkarībā no tā, kuru pogu lietotājs nospiedis. Ja neviena no pogām nav nospiesta, tad iepriekšējais spožuma pieaugums netiek mainīts.
- Pēc tam mainām mirdzdiodes spožumu atbilstoši noteiktajam spožuma pieaugumam.
- Nepieciešamas vēl papildu pārbaudes, lai jaunā spožuma vērtība tiktu noturēta iepriekš minētā intervāla robežās.
- Cikla beigās tiek noteikta neliela aizture, lai varētu novērot spožuma izmaiņas,

Interesanti! *Tie, kam uzdevums šķiet vienkāršs, var papildināt programmu tā, lai, nospiežot abas pogas vienlaicīgi, mirdzdiodes spožums kļūtu nemainīgs.*

Shēma



3.4. DARBA LAPA. Mirdzdiodes mirgošanas ātruma izmaiņas

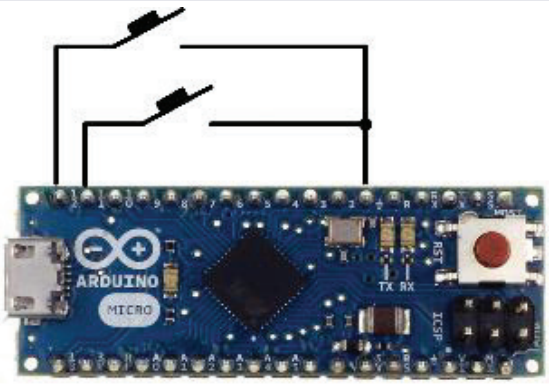
Mērķis

Izveidot programmu, kas ļauj ar divām pogām regulēt mirdzdiodes spožuma izmaiņas ātrumu, t. i., ar vienu pogu to iespējams samazināt, bet ar otru — palielināt.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Montāžas vads	2 gab.

Izveidojamā shēma



Šāda shēma ļauj izveidot ķēdi, kas nodrošina mirdzdiodes spožuma regulēšanu. Kā vienmēr, pati svarīgākā ir programmas izpildes loģika. Konstruktorā pogas **P1** un **P2** ir pieslēgtas pie kopējā izvada — 0 pola. Tas nozīmē, ka reālo shēmu var vienkāršot tā, kā tas norādīts attēlā "Nepieciešamie materiāli un slēgums", t. i., nav jāizmanto papildu detaļas.

Darba izpildes soļi

1. Izveidojiet slēguma shēmu.
2. Sastādiet doto programmu.
3. Ierakstiet programmu MK Arduino atmiņā tā, kā tas tika norādīts programmas izstrādes vides sagatavošanas nodaļā.
4. Izbaudiet paša izveidotā spožuma mainītāja darbību.

```
int ledKaja = 13;
int laiks = 500;

void setup()
{
   pinMode(ledKaja,OUTPUT);
   pinMode(11,INPUT);
   pinMode(12,INPUT);
}
void loop()
{
   bool PogasStavoklis_1 = digitalRead(12);
      //Nolasām 12. MK kājas stāvokli
   bool PogasStavoklis_2 = digitalRead(11);
      //Nolasām 11. MK kājas stāvokli
   if(PogasStavoklis_1 == true)
                        { laiks = laiks + 50; }
   if(PogasStavoklis_2 == true)
                        { laiks = laiks - 50; }
   if(laiks >= 1000) { laiks = 1000; }
   if(laiks < 0) { laiks = 0; }
   digitalWrite(ledKaja, HIGH);
   delay(laiks);
   digitalWrite(ledKaja, LOW);
   delay(laiks);
}
```

Īss skaidrojums

Globālie mainīgie:

- Tiek definēti divi mainīgie:
  - **ledKaja** – norāda uz MK iebūvētu mirdzdiodi.
  - **laiks** – laiks milisekundēs, ar kuru tiek mainīts mirdzdiodes stāvoklis.

Inicializācijas daļa:

- Līdzīgi kā līdz šim, tiek noteiktas ieejas un izejas.

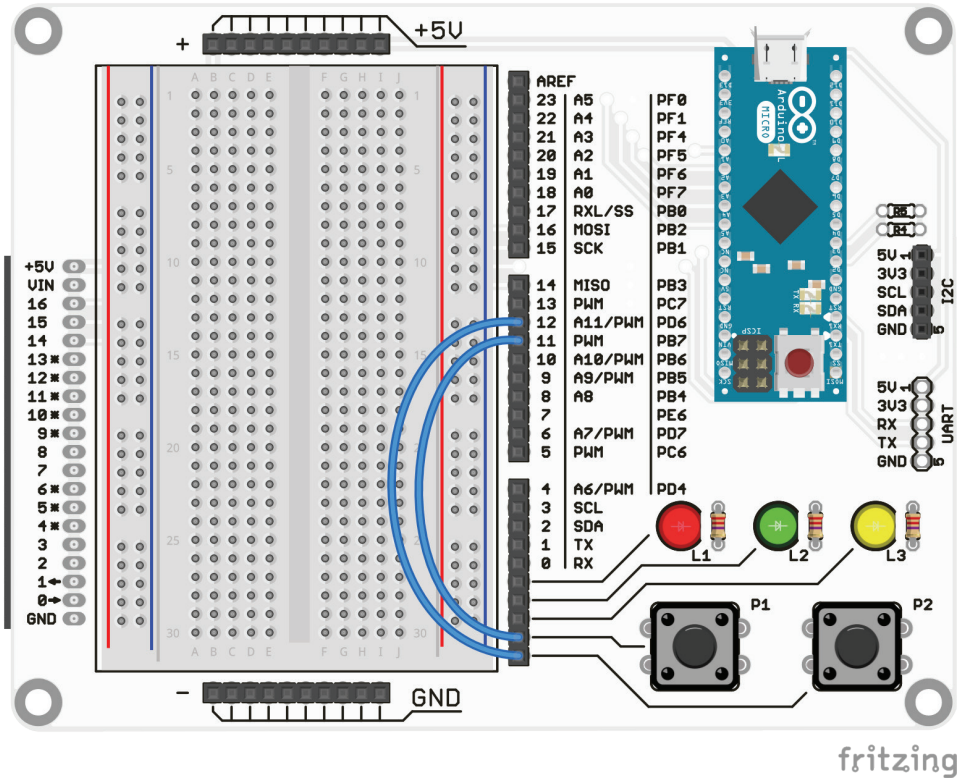
Cikla daļa:

- Ir definēti divu pogu stāvokļi: Pogas_Stavoklis_1 un Pogas_Stavoklis_2, kas saturēs atbilstoši 11. un 12. MK kājas stāvokli viena cikla ietvaros.
- Ir ieviesti nosacījumi, kuri ļauj regulēt mirdzdiodes spīdēšanas un izdzišanas periodu, kas tiek mainīti ar soli 50 milisekundes.

- Nepieciešamas vēl papildu pārbaudes, lai jaunā laika vērtība tiktu noturēta intervālā 0–1000 milisekundes.
- Pēc tam tiek ieslēgta un izslēgta mirdzdiode ar jauno laika aiztures vērtību.

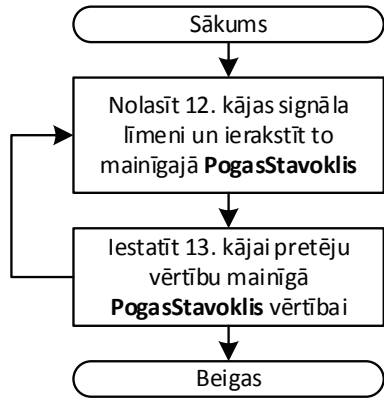
Interesanti! *Ja uzdevums šķiet vienkāršs, var papildināt programmu tā, lai, nospiežot abas pogas vienlaicīgi, mirdzdiodes ieslēgšanas un izslēgšanas intervāls nemainās.*

Shēma

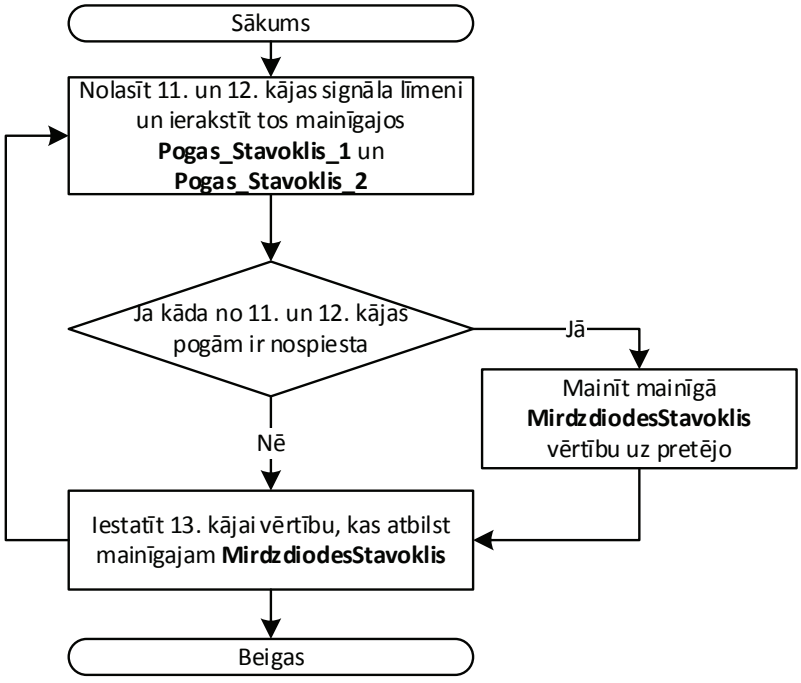


3. nodaļas pielikums

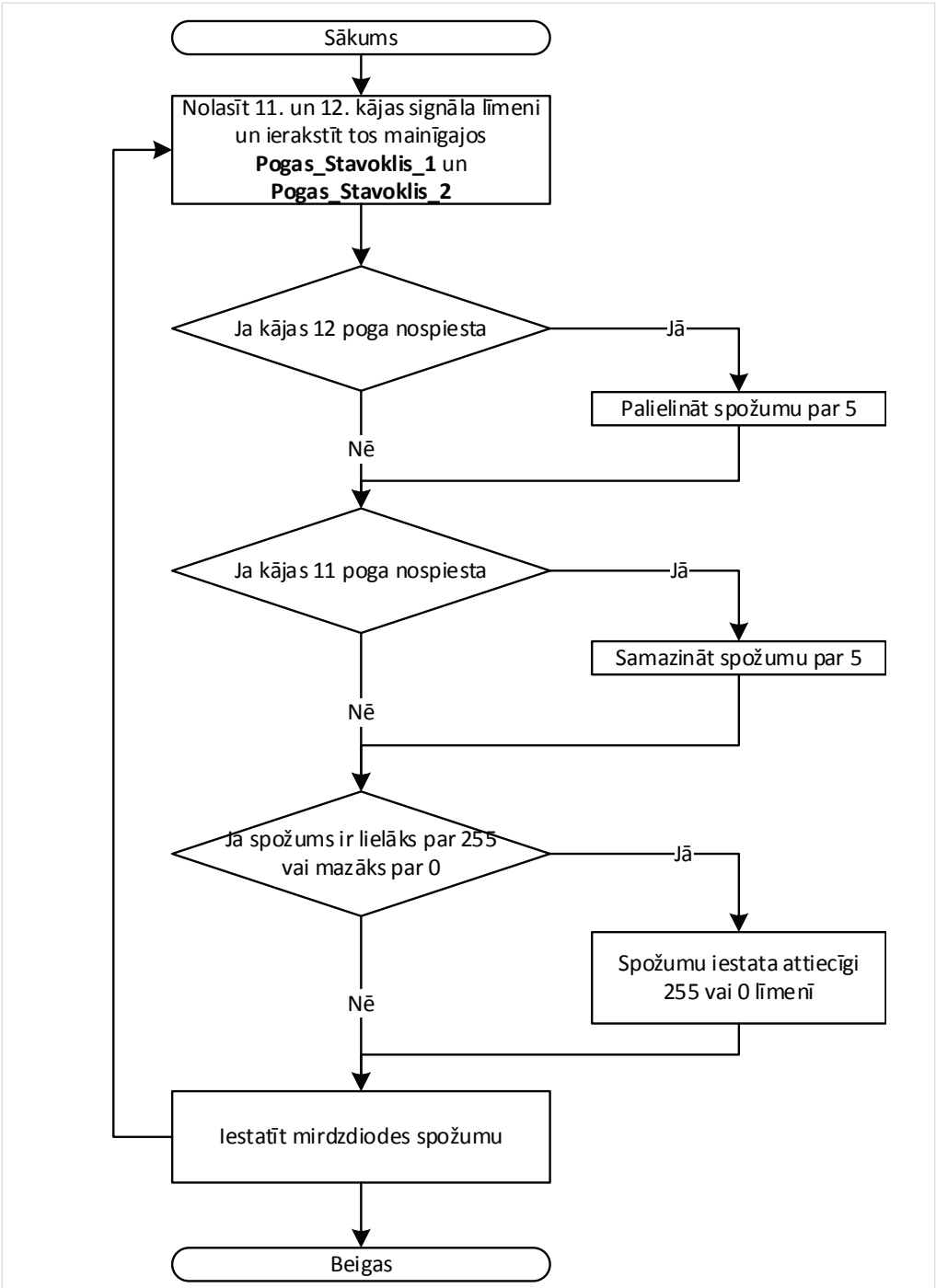
Varat papētīt šajā nodaļā realizēto programmu blokshēmas.



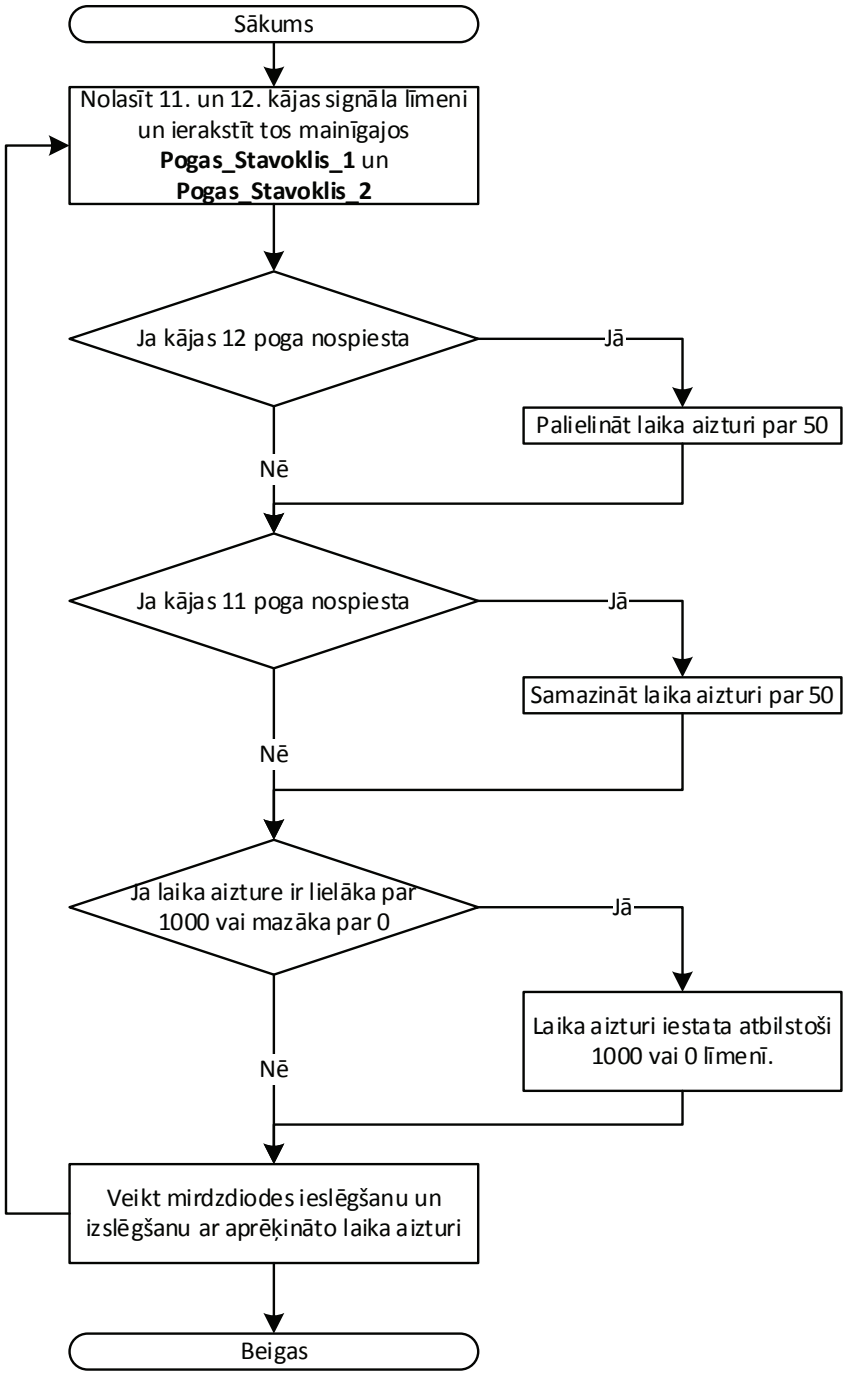
1. uzdevums — mirdzdiodes ieslēgšanas blokshēma.



2. uzdevums — kāpņutelpas apgaismojuma sistēmas blokshēma.



3. uzdevums — mirdzdiodes spožuma blokshēma.



4. uzdevums — mirdzdiodes mirgošanas ātruma izmaiņas.

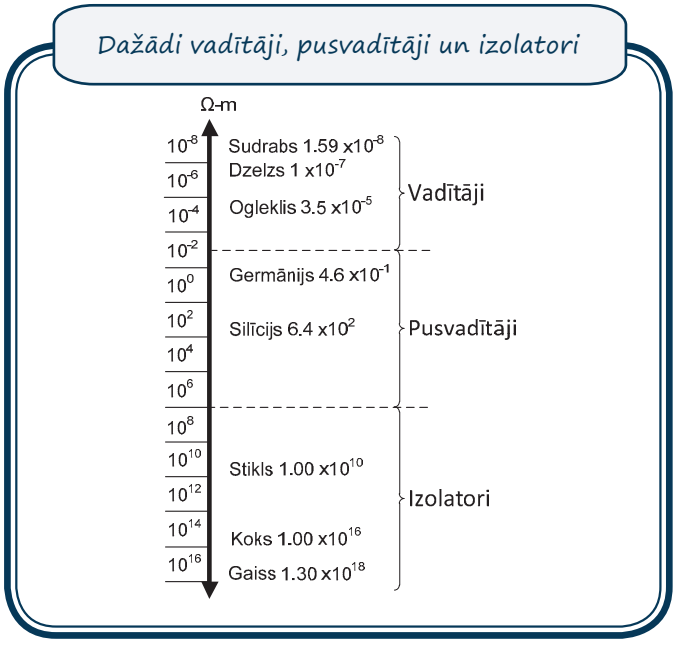
4. PUSVADĪTĀJI

4.1. Mērķis

Temata mērķis ir iepazīstināt ar elektronikā biežāk izmantotajiem pusvadītāju elementiem, kas nodrošina dažādu uzdevumu izpildīšanu. Pusvadītāji ļauj, piemēram, izveidot elektronisku slēdzi, kas savukārt var nodrošināt kādu konkrētu iekārtas funkciju, piemēram, tās ieslēgšanu/izslēgšanu.

4.2 Teorētiskā daļa

Pusvadītāji nav sevišķi labi elektrības vadītāji un nav arī sevišķi labi izolatori, no tā arī cēlies nosaukums pusvadītāji. Pusvadītāji ir silīcijs (Si) un germānijs (Ge).



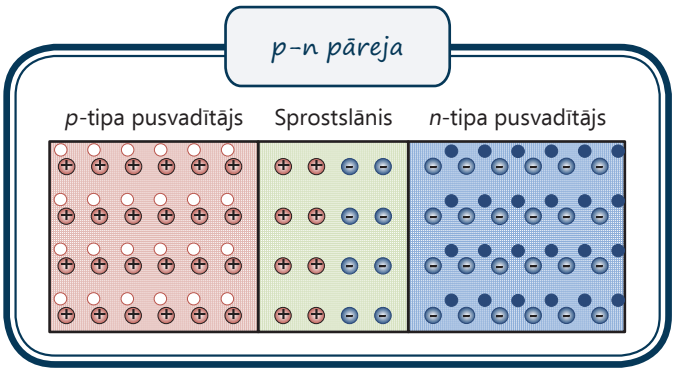
Lai palielinātu pusvadītāju spēju vadīt elektrību, tiem piejauc klāt dažādus piemaisījumus — donorus un akceptorus. Ja piejauksim donorus, tad iegūsim negatīvi lādētu *n* tipa pusvadītāju, kurā būs daudz elektrību nesošo elektronu. Ja piejauksim akceptorus, tad iegūsim pozitīvi lādētu *p* tipa pusvadītāju, kurā būs daudz caurumu, ko aizpildīt elektroniem.

n un *p* tipa pusvadītāji tiek izmantoti, lai ražotu tādus elektronikas elementus kā diodes, tranzistorus, saules baterijas, mikroshēmas u. c. Bez pusvadītājiem nebūtu iedomājamas lielākā daļa mūsdienu ierīču.

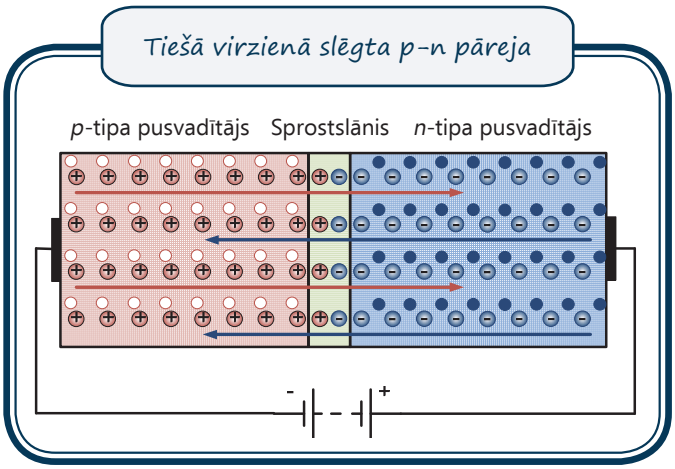
4.3. Diodes

4.3.1. *p-n* pāreja

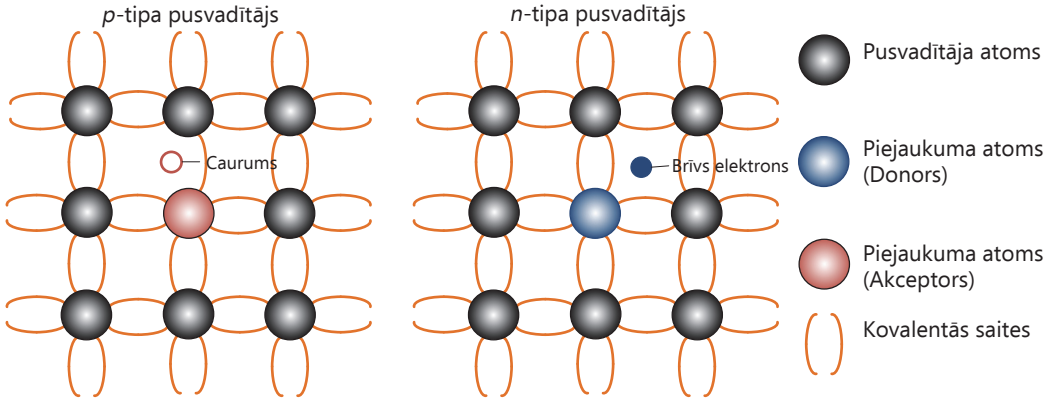
Savienojot *p* un *n* tipa pusvadītājus, iegūst *p-n* pāreju. Kā iepriekš noskaidrojām, *n* tipa pusvadītājā ir daudz elektronu, bet *p* tipa pusvadītājā ir daudz caurumu. Savienojot *p* un *n* tipa pusvadītājus, elektroni no *n* reģiona sāk pārvietoties uz caurumiem *p* reģionā, neitralizējot viens otru. Šis process turpinās, līdz vietā, kur saduras divis materiāli, tiek izveidots sprostslānis, kurš neļauj elektroniem nokļūt pie caurumiem.



Ja mēs pieliksim papildu sprieguma avotu *p-n* pārejai, kur *p* tips būs pievienots pie pozitīvā gala un n tips pie negatīvā, un spriegums būs vismaz 0,7 V, sprostslānis samazināsies un caur *p-n* pāreju sāks plūst elektroni. To sauc par tiešā virzienā slēgtu *p-n* pāreju.

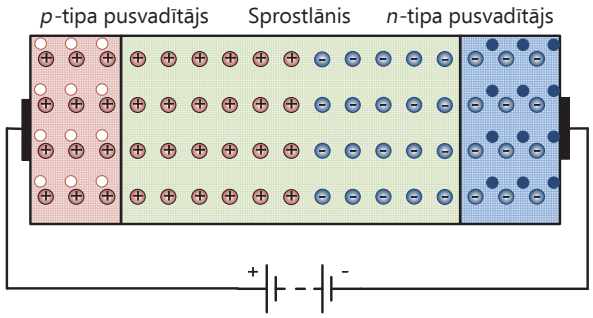


n tipa un p tipa pusvadītāju struktūra



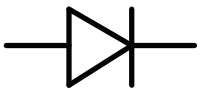
Ja pieliksim papildu sprieguma avotu *p-n* pārejai, kur p tips būs pievienots pie negatīvā gala un n tips pie pozitīvā gala, sprostslānis palielināsies, un strāva neplūdīs caur šo slāni. To sauc par reversā virzienā slēgtu *p-n* pāreju.

Reversā virzienā slēgta p-n pāreja



4.3.2. Diodes īpašības

Vienkāršākā ierīce, ko var izveidot, saliekot kopā divus *p* un *n* tipa pusvadītājus, ir diode. Tās galvenā īpašība ir spēja laist strāvu tikai vienā virzienā.

Apzīmējums	Shēmas simbols
	Diodi apzīmē ar bultiņu, kuras galā ir taisna līnija. Bultiņa norāda, kurā virzienā diode vada strāvu.

Diode sāk vadīt strāvu, kad tās anods ir vismaz par 0,7 voltiem pozitīvāks nekā katods. Tas nozīmē — ja spriegums uz diodes būs mazāks par 0,7 voltiem, diode strāvu nevadīs. Tā kā diode ir aktīvā komponente, tad tā arī turpinās uzturēt 0,7 voltu spriegumu starp saviem izvadiem, kamēr strāva caur diodi strauji pieaugs. Diode ir paredzēta tikai noteiktam strāvas daudzumam, tāpēc, ja tiek pārsniegts paredzētais daudzums, diode var sākt karst un beigās sadegt.

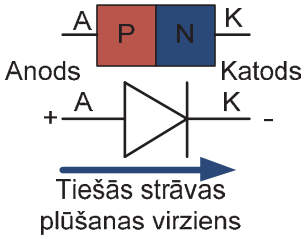
Saslēdzot diodi reversā virzienā, kad spriegums uz katoda ir pozitīvāks nekā uz anoda, neļaus plūst strāvai caur diodi. Katra diode ir paredzēta, lai izturētu tika noteiktu sprieguma daudzumu reversā virzienā. Ja tiks pārsniegts šis spriegums, diode tiks caursista, tā sāks vadīt strāvu un tiks bojāta.

4.3.3. Diožu parametri, veidi, pielietojumi

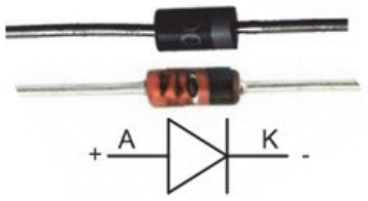
Ir ārkārtīgi daudz dažādu diožu veidu un pielietojumu, bet mēs apskatīsim vienkāršākās signāla diodes parametrus veidus un pielietojumus.

Signāla diodes varbūt dažādas, kopīgs tām ir pielietojuma veids, bet atšķirīgi ir to parametri un izskats. Visbiežāk satapsiet divu veidu signāla diožu korpusus — melnus ar pelēku svītru galā vai sarkana stikla ar melnu svītru galā. Svītras uz diožu korpusiem parasti norāda katoda atrašanās vietu, pēc tā var noteikt, kā diode ievietojama shēmā.

Diodes apzīmējums un uzbūve



Biežāk sastopamie signāla diožu korpusi



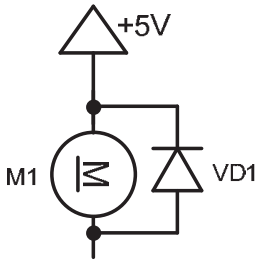
Diodes atšķiras pēc to parametriem. Diodes parametrus var noskaidrot, apskatot ražotāja sniegto tehnisko specifikāciju. Ierīces ražotājs katrai elektroniskai komponentei ir izveidojis sava datu lapu, kur pastāstīts viss, kas jāzina ierīces lietotājam. Lai atrastu diodes datu lapu, ir jāzina diodes nosaukums, parasti tas ar maziem burtiņiem uzdrukāts uz diodes korpusa. Ievadot šo numuru interneta pārlūkprogrammā un pierakstot klāt "datasheet", var atrast diodes datu lapu. Lielākā daļa datu lapu ir angļu valodā. Diodes raksturo daudz un dažādi parametri, bet šeit apskatīsim divus svarīgākos:

- **Maksimālā tiešā virziena strāva** I_F nosaka to, cik stipra drīkst būt strāva, kas plūst caur diodi. Diodei ir neliela pretestība uz p-n pārejas, tādējādi pēc Oma likuma izdalās jauda siltuma veidā. Ja tiek pārsniegts pieļaujamais strāvas stiprums, diode pārkarst un sadeg. Lai ierobežotu strāvu, rezistorus bieži lieto virknē ar diodi.
- **Maksimālais inversais spriegums** U_R nosaka to, cik lielu spriegumu drīkst pielikt uz diodes izvadiem inversā virzienā, pirms notiek diodes caursite. Ja šis spriegums tiek pārsniegts, diode tiek caursista, sāk vadīt strāvu un tiek bojāta.

Atkarībā no parametriem diodēm ir vairāki pielietojumi:

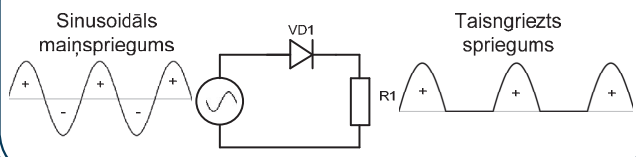
- **Komponentu aizsardzība.** Diodes bieži izmanto komponentu aizsardzībā, pret dažādiem inversiem spriegumiem un strāvām. Ar diodi shēmu var aizsargāt, ja baterija tiek ielikta nepareizi, jo diode ļauj strāvu vadīt tikai vienā virzienā. Tā pasargā shēmu arī no elementiem, kuru sastāvā ir induktīvi elementi (spoles, motori). Spoles pretojas strāvas izmaiņām, un, ja izslēdzam motorus, tās var radīt bīstami lielas strāvas, kas ir vērstas pretēji normālam strāvas plūšanas virzienam. Ieliekot diodi, šī strāva plūst caur pašu spoli un nebojā pārējo shēmu.

Diodes aizsardzības izmantošana motorā



- **Līdzstrāvas iegūšana.** Diodes izmanto, lai pārveidotu maiņstrāvu līdzstrāvā, šo procesu sauc par taisngriešanu. Maiņstrāvas polaritāte regulāri mainās, diode ļauj tai plūst tika vienā virzienā, pārveidojot to līdzstrāvā.

Maiņstrāvas pārveidošana līdzstrāvā

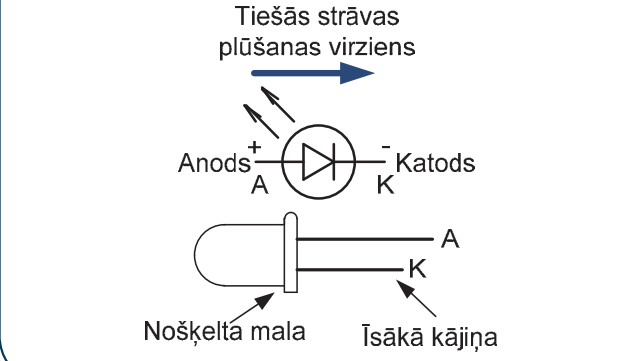


4.4. Mirdzdiodes

Iepriekšējās shēmās jau apskatījām mirdzdiodes. Mirdzdiodes ir īpašs diožu veids, kas atšķirībā no parastajām diodēm izstaro gaismu. Mirdzdiodēm ir pavisam savādāks korpuss, kuru veido caurspīdīga plastmasa, kas aizsargā diodi un ļauj tai cauri laist gaismu. Tāpat kā parastā diode arī mirdzdiode laiž strāvu cauri tikai vienā virzienā, tāpēc ir svarīgi zināt, kā pareizi diodi ieslēgt shēmā. Ir divas drošas pazīmes, kā noteikt diodes virzienu:

- Katoda pusē diodes mala ir nošķelta;
- Anoda kāja parasti ir garāka par katoda kāju.

Mirdzdiodes katoda un anoda noteikšana



Mirdzdiode ir viens no labākajiem gaismas avotiem. Atšķirībā no kvēlspuldzēm tās lielāko daļu enerģijas pārveido gaismā, nevis siltumā, tās ir izturīgas, darbojas daudz ilgāk un var izgatavot mazākā izmērā.

Mirdzdiodes krāsu nosaka izmantotais pusvadītāja materiāls. Ja parastajās diodēs izmanto silīciju, tad gaismas diodēs izmanto tādus elementus kā gallija fosfātu, silīcija karbīdu u. c. Tā kā izmantotie pusvadītāji ir dažādi, atšķiras arī spriegums, kas nepieciešams, lai mirdzdiode sāktu spīdēt. Tabulā ir parādīts, ar kādiem pusvadītājiem var iegūt dažādas krāsas un kāds spriegums ir nepieciešams diodes iedarbināšanai.

Mirdzdiožu parametri izolatoriem

Diodes parametri			
Pusvadītājs	Viļņa garums (nm)	Krāsa	Spriegums (V) uz mirdzdiodes U_D (20 mA)
GaAs	850–940	Infrasarkanā	1,2
GaAsP	630–660	Sarkana	1,8
GaAsP	605–620	Oranža	2,0
GaAsP:N	585–595	Dzeltena	2,2
AlGaP	550–570	Zaļa	3,5
SiC	430–505	Zila	3,6
GaInN	450	Balta	4,0

Kad mirdzdiode tiek pieslēgta spriegumam un tā ieslēdzas, caur to sāk plūst ļoti liela strāva, kas var bojāt diodi. Tāpēc **visām mirdzdiodēm klāt jābūt strāvu ierobežojošam rezistoram.**

Strāvu ierobežojošā rezistora pretestību nosaka trīs parametri:

- Strāva, kas drīkst plūst cauri mirdzdiodei — I_D;
- Spriegums, kas nepieciešams, lai mirdzdiode sāktu darboties — U_D;
- Kopējais spriegums uz mirdzdiodes un rezistora — U.

Lai aprēķinātu diodei nepieciešamo pretestību, jārīkojas šādi:

1. Jānoskaidro spriegums, kas nepieciešams mirdzdiodes darbībai U_D, to var atrast mirdziožu parametru tabulā.
2. Jānoskaidro strāvas stiprums, kas nepieciešama, lai mirdzdiode spīdētu I_D. To var sameklēt mirdzdiodes datu lapā, bet, ja nevar atrast, tad 20 mA strāva parasti ir pareiza un droša izvēle.
3. Jānoskaidro kopējais spriegums uz mirdzdiodes un rezistora, parasti tas ir shēmas barošanas spriegums U.
4. Visi lielumi jāievieto šādā formulā:

$$R = \frac{U - U_D}{I_D}$$

5. Iegūst pretestību rezistoram, lai mirdzdiodi varētu lietot droši.
6. Atrod rezistora nominālu, kas atbilst vai ir lielāks par aprēķināto.

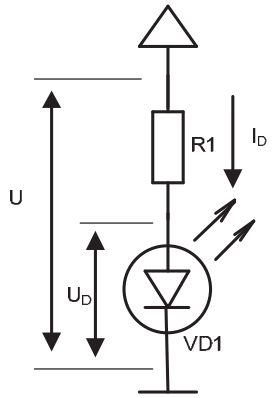
Piemērs

Shēmā slēgsim oranžu mirdzdiodi. Oranžai mirdzdiodei ir nepieciešams $U_D = 2$ V spriegums. Strāva, kas plūdīs cauri diodei, būs $I_D = 20$ mA. Shēmas spriegums būs $U = 5$ V. Ievietojam visus skaitļus formulā:

$$R = \frac{U - U_D}{I_D} = \frac{5 - 2}{0,02} = \frac{3}{0,02} = 150\ \Omega$$

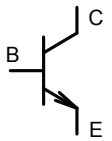
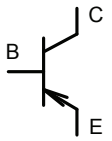
Tātad — lai ieslēgtu oranžu mirdzdiodi, ir nepieciešams rezistors ar 150 Ω vai lielāku pretestību.

Mirdzdiodes pieslēgšanas shēma



4.5. Tranzistori

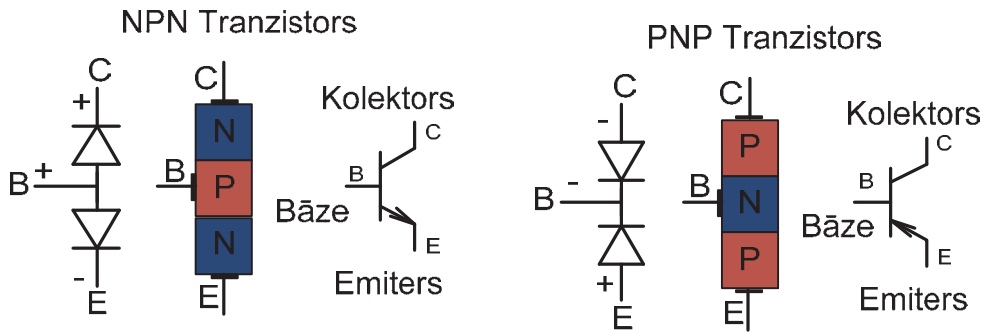
Saliekot kopā divas *p-n* pārejas, iegūst bipolāro tranzistoru. Tranzistors parasti ir ierīce ar trīs izvadiem: kolektoru (C), bāzi (B) un emiteru (E). Tranzistori pilda divas funkcijas: tie kalpo kā bezkontaktu slēdzis un kā sprieguma un strāvas pastiprinātājs.

Apzīmējums	Shēmas simbols
	NPN tranzistors — emitera bultiņa vērsta prom no bāzes, kas norāda, ka signāla strāva izplūst no bāzes uz emiteru.
	PNP tranzistors — emitera bultiņa vērsta uz bāzi, kas norāda, ka signāla strāva plūst uz bāzi no emitera.

4.5.1. Tranzistoru uzbūve

Tranzistors tiek izgatavots, pievienojot pārejas diodei papildu pusvadītāju kārtu *p-n*. Tāpēc tranzistorus var iztēloties kā divas kopā saliktas diodes. Bipolārais tranzistors sastāv no divām *p-n* pārejām, izveidojot trīs izvadus: kolektoru, bāzi un emiteru. *p-n* pārejas var salikt divos veidos: *n-p-n* un *p-n-p*, tādējādi ir divu veidu tranzistori — NPN un PNP. Abiem tranzistoru tipiem darbības princips ir vienāds, atšķiras tranzistora pieslēgšana shēmā un barošanas polaritātes.

PNP un NPN tranzistoru uzbūves salīdzinājums



4.5.2. Darbība PNP un NPN

Tranzistors ir ierīce, kuru vada ar strāvas palīdzību. Bāzes strāvas lielums nosaka tranzistora stāvokļus. Ja tranzistora bāzē nav strāvas, tranzistors būs izslēgts. Ja tranzistora bāzē ir neliela strāva, tranzistors būs ieslēgts un darbosies kā pastiprinātājs. Jo lielāka strāva tranzistora bāzē, jo lielāka strāva plūdīs starp kolektoru un emiteru. Ja bāzē plūst pietiekami liela strāva, tad tranzistors ir pilnībā atvērts, un starp kolektoru un emiteru plūst strāva, līdzīgi kā ieslēgtā slēdzī.

Tranzistors var būt:

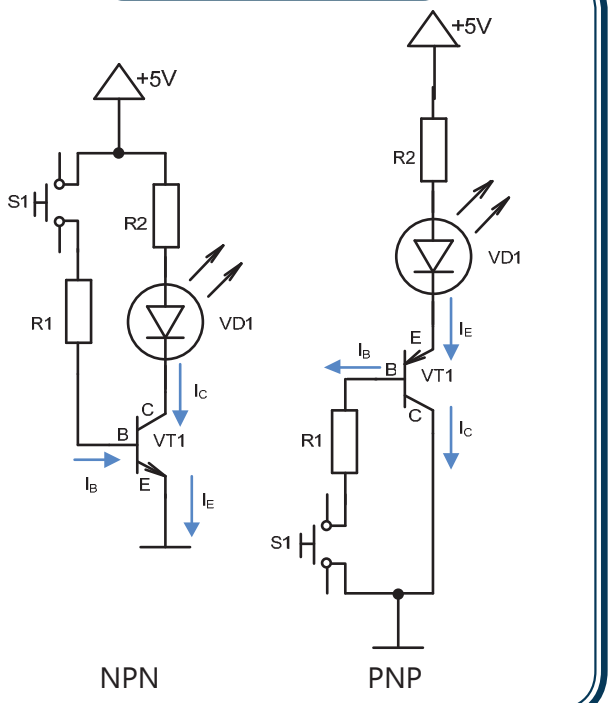
- Izslēgts — strāva starp kolektoru un emiteru neplūst;
- Aktīvs — tranzistors darbojas kā pastiprinātājs;
- Piesātināts — tranzistors ir pilnībā atvērts un darbojas kā ieslēgts slēdzis.

Lai varētu ieslēgt NPN tranzistoru, tā bāzei jābūt ar pozitīvāku spriegumu nekā emiteram, tāpēc bāzi pieslēdz pie pozitīvā sprieguma. PNP tranzistoram ir otrādi — emiteram ir jābūt ar pozitīvāku spriegumu nekā bāzei, tāpēc emiters tiek pieslēgts pie pozitīvā izvada, bet bāze pie negatīvā. Attiecīgi arī mainās strāvas plūšanas virzieni. Tranzistora bāzē drīkst plūst ļoti neliela strāva, tāpēc ir **svarīgi lietot strāvu ierobežojošo rezistoru *R1*, lai tranzistors netiktu bojāts.**

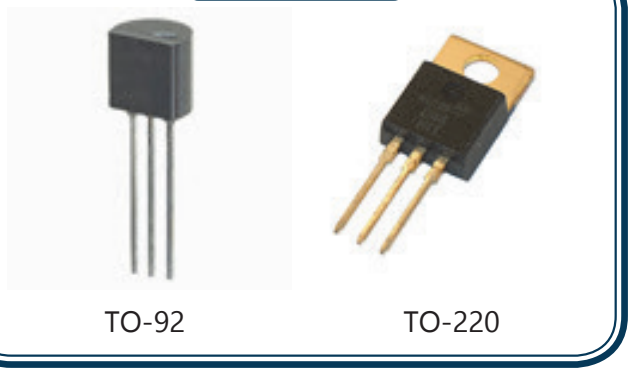
4.5.3. Tranzistora korpuss un parametri

Tāpat kā diodēm arī tranzistoru korpusiem ir atšķirīgi parametri. Vieni no populārākajiem ir TO-92 un TO-220. Var redzēt, ka šie korpusi atšķiras pēc izmēriem un izskata. Korpusa izmērus nosaka nepieciešamība izkliedēt jaudu, lai tranzistora lietošanas laikā tas nepārkarstu un nesadegtu. Jo lielāks tranzistora korpuss, jo vairāk jaudas tas var izkliedēt un jo lielāku strāvu tas var vadīt.

Tranzistoru lietošana

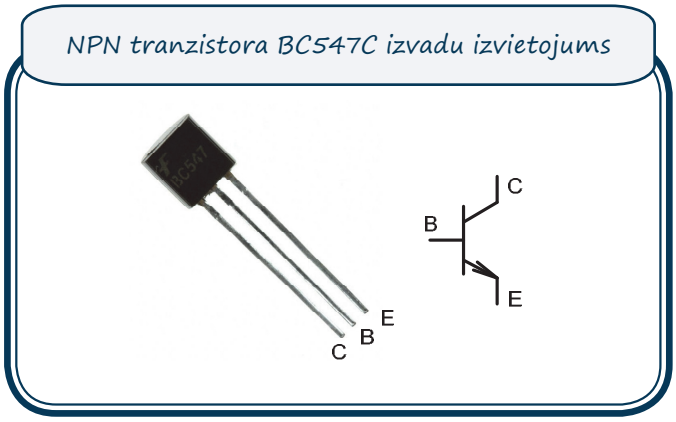


Tranzistori



Lietojot tranzistorus, ir svarīgi zināt tā izvadu izvietojumu (kolektora, bāzes, emitera). Tomēr katram tranzistoram tas var atšķirties, tāpēc svarīgi izpētīt tranzistora datus. Līdzīgi kā diodēm, arī tranzistora nosaukumu var atrast uz ierīces korpusa — ierakstot to interneta meklētājā, var atrast tā datu lapu, kurā būs norādīts kāju izvietojums.

NPN tranzistora BC547C izvadu izvietojums



Lietojot tranzistorus, ir svarīgi zināt vairākus parametrus, kuri nosaka tranzistora darbību:

1. **Maksimāli pieļaujamais kolektora emitera spriegums U_{CE0}** — nosaka to, cik lielu spriegumu drīkst izmantot tranzistorā, lai tas netiktu bojāts.
2. **Maksimāli pieļaujamā kolektora strāva I_C** — nosaka to, cik liela strāva drīkst plūst caur tranzistoru, lai tas netiktu bojāts. Pārsniedzot šo robežu, tranzistors pārkarst un sadeg.
3. **Strāvas pastiprināšanas koeficients β** — nosaka to, cik reižu mainīsies kolektora strāva, mainot bāzes strāvu $I_C = I_B\beta$.
4. **Maksimāli pieļaujamais kolektora bāzes spriegums U_{CBO}** ietekmē, cik liela bāzes strāva I_B plūdīs caur tranzistoru, tāpēc, lai nesabojātu tranzistoru, šī vērtība jāsaglabā zem maksimālās.

Ievērojot ražotāja noteiktos maksimālos parametrus, tiek nodrošināta tranzistora ilgmūžība.

4.1. DARBA LAPA. Diodes slēgums

Mērķis

Novērot diodes pamatīpašību — strāvas laišanu vienā virzienā.

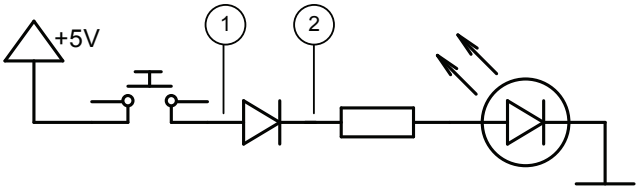
Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā "Diode slēgta strāvas plūšanas virzienā".
2. Pieslēgt shēmu sprieguma avotam un nospiest pogu.
3. Novērot, vai mirdzdiode spīd.
4. Nomērīt spriegumu uz diodes, salīdzināt šo spriegumu ar teorētiski aprakstīto spriegumu, kas krīt uz diodes. Kā tas atšķiras?
5. Saslēgt shēmu, kas redzama attēlā "Diode slēgta pretēji strāvas plūšanas virzienam" (diode maina virzienu):
6. Pieslēgt shēmu sprieguma avotam un nospiest pogu.
7. Novērot, vai mirdzdiode nespīd.

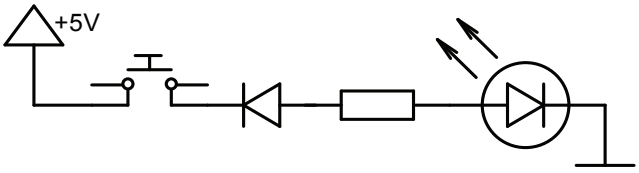
Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Poga	1 gab.
Diode PH4148	1 gab.
Rezistors (330 Ω)	1 gab.
Mirdzdiode — sarkana	1 gab.
Montāžas vads	3 gab.

Izveidojamās shēmas

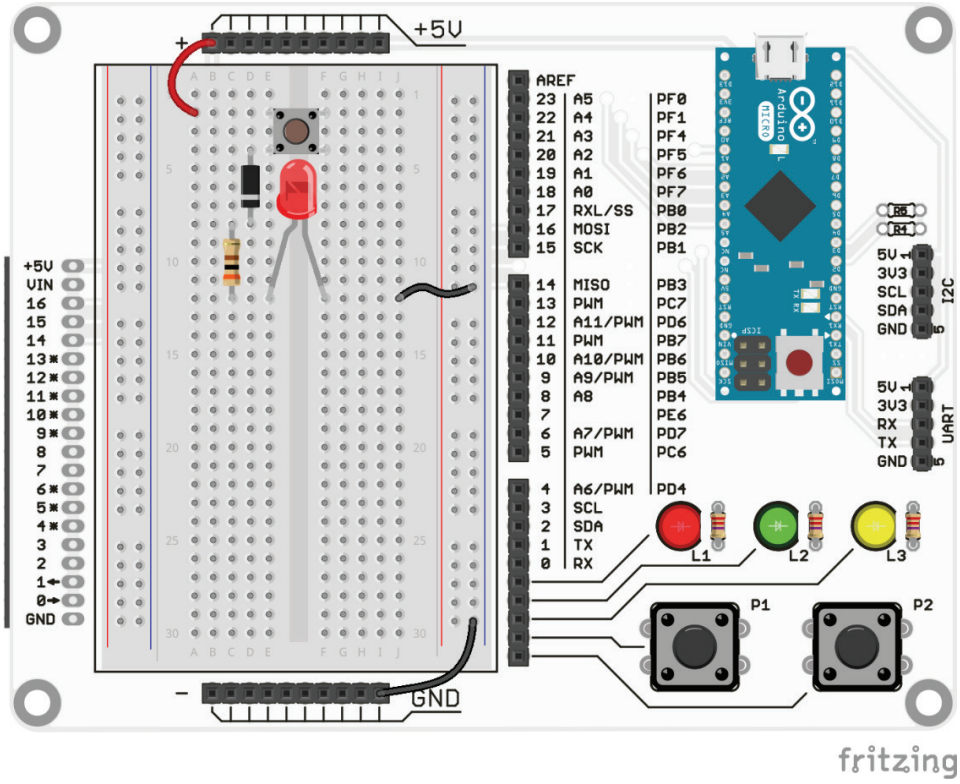


Virknē saslēgti elementi ar diodi strāvas plūšanas virzienā.

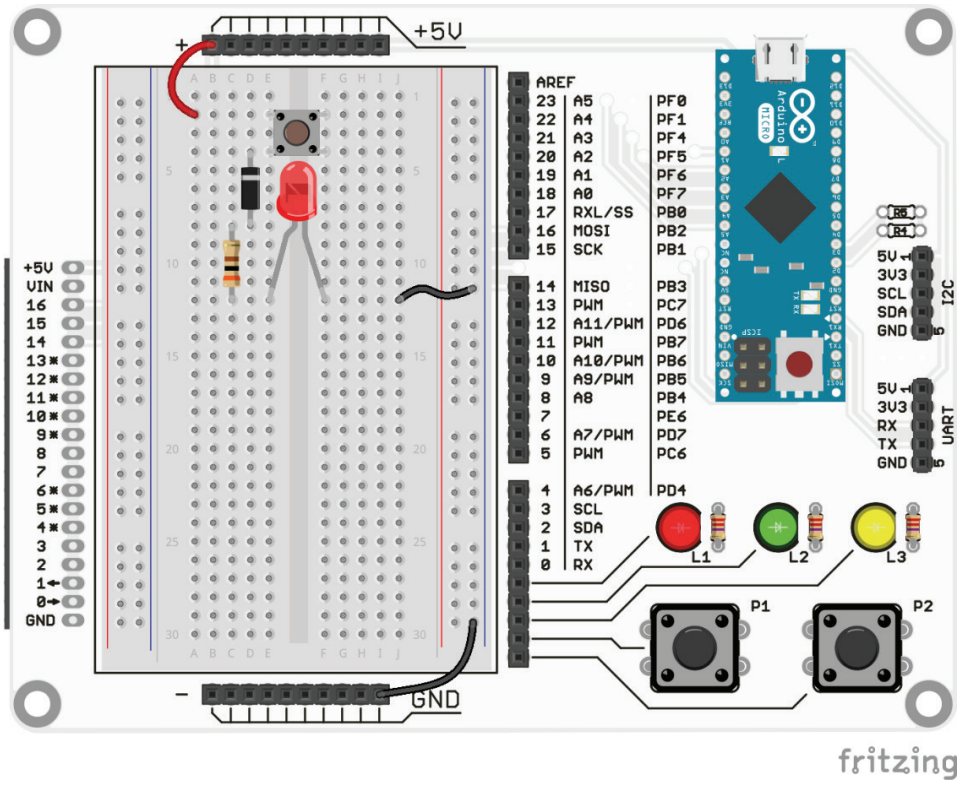


Virknē saslēgti elementi ar diodi pretēji strāvas plūšanas virzienam.

Diode slēgta strāvas plūšanas virzienā



Diode slēgta pretēji strāvas plūšanas virzienam



4.2. DARBA LAPA. Mirdzdiodes spriegums

Mērķis

Novērot atšķirību starp dažādām mirdzdiodēm.

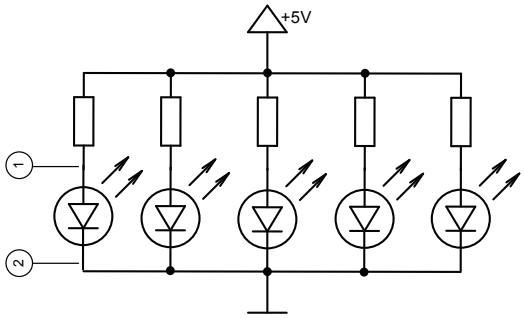
Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā:
2. Pieslēgt shēmu sprieguma avotam.
3. Novērot, vai katra mirdzdiode spīd ar atšķirīgu spožumu.
4. Izmantojot multimetru līdzsprieguma mērīšanai, izmērīt spriegumu uz katras mirdzdiodes, starp punktiem **1** un **2**.
5. Salīdzināt ar teorētiskajiem mirdzdiožu spriegumiem. Kāda ir atšķirība?

Nepieciešamie materiāli

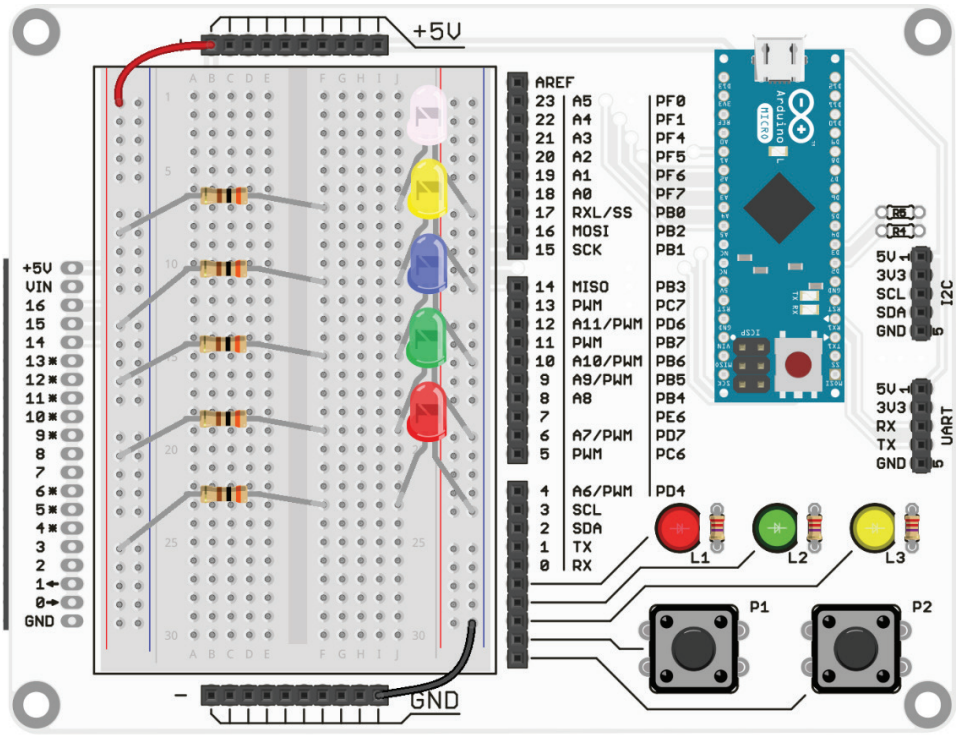
Materiāls/detaļa	Skaits
Rezistori (330 Ω)	5 gab.
Mirdzdiode — balta, dzeltena, zila, zaļa, sarkana	5 gab.
Montāžas vads	Vairāki

Izveidojamā shēma



Paralēli saslēgtas vairāku krāsu mirdzdiodes.

Mirdzdiožu slēgums

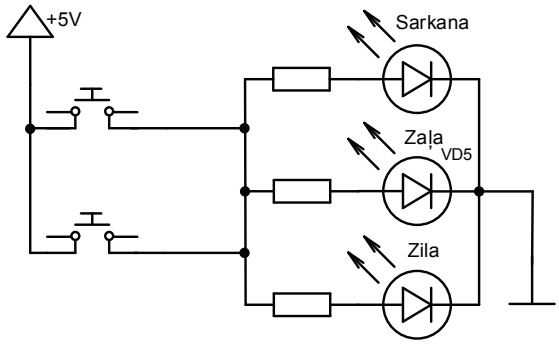


4.3. DARBA LAPA. RGB mirdzdiode

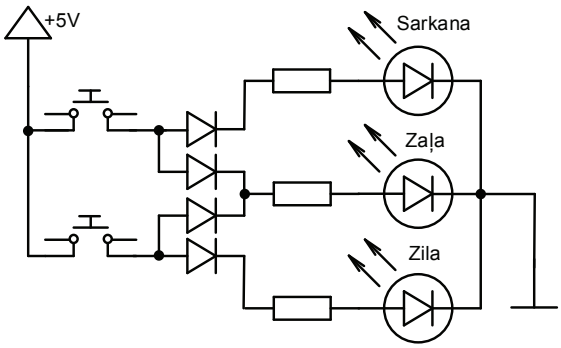
Mērķis

Apgūt RGB LED izmantošanu un diožu pamatīpašību.

Izveidojamās shēmas



Nospiežot jebkuru pogu RGB, diode degs ar baltu gaismu.



Nospiežot vienu vai otru pogu, diode degs ar dzeltenu vai gaiši zilu gaismu.

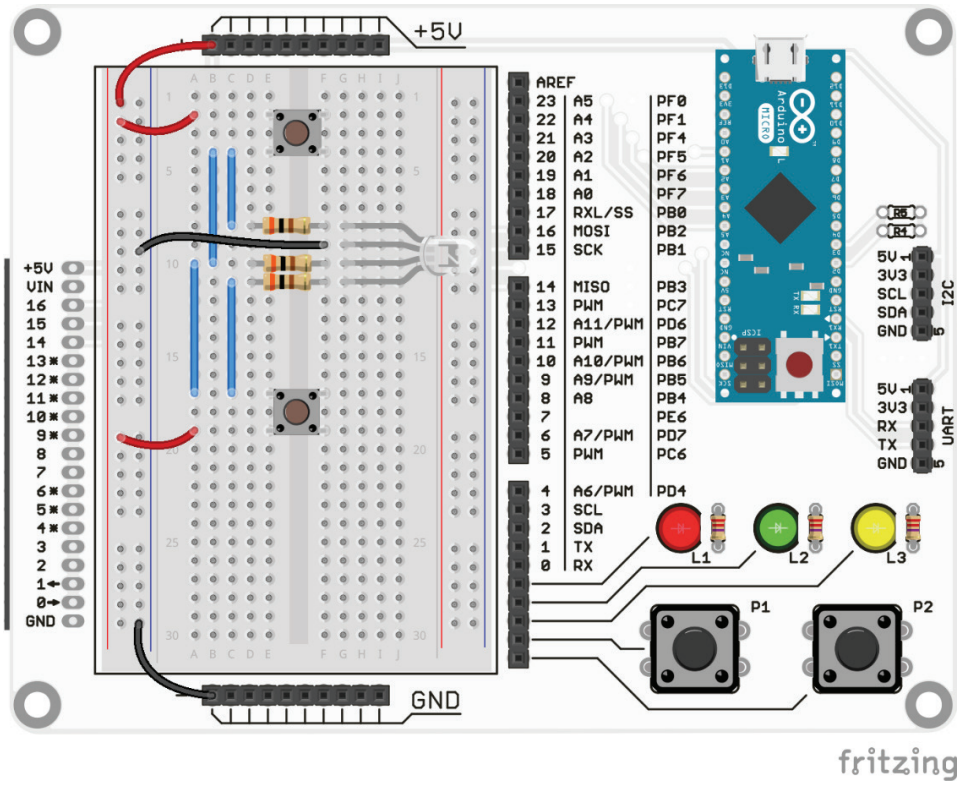
Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Diode PH4148	4 gab.
Poga	2 gab.
Rezistors (330 Ω)	3 gab.
RGB mirdzdiode	1 gab.
Montāžas vads	Vairāki

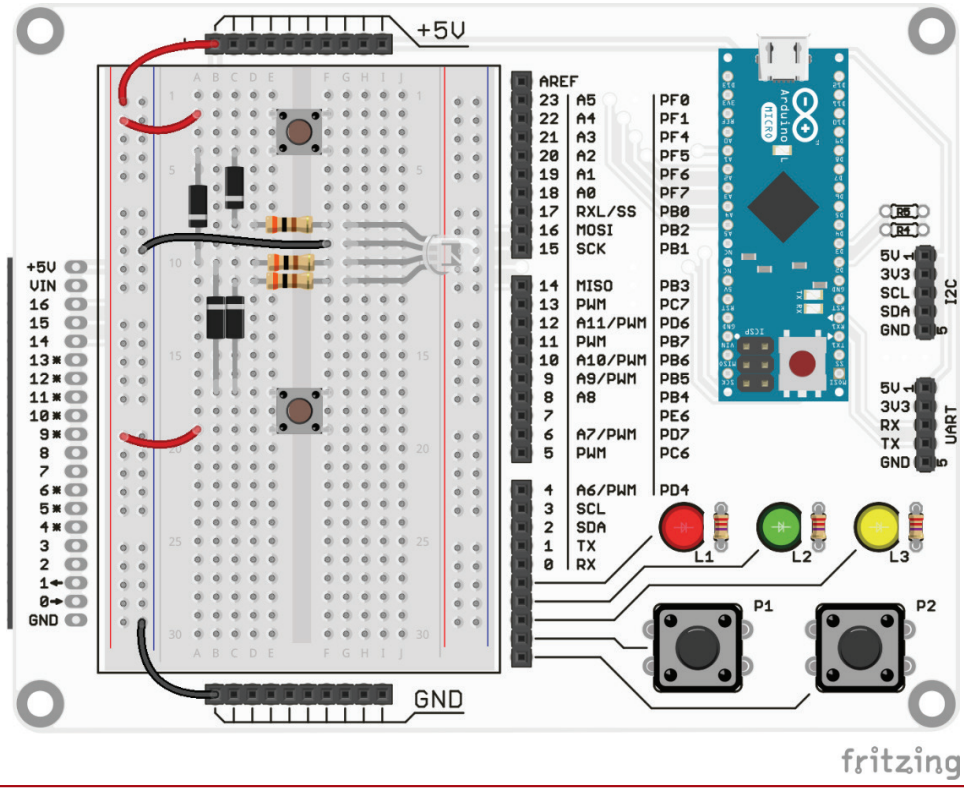
Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā "RGB diodes vadība bez diodēm".
2. Pieslēgt shēmu sprieguma avotam.
3. Nospiest vispirms vienu pogu, pēc tam otru.
4. Novērot, vai pēc abu pogu nospiešanas RGB mirdzdiode deg ar baltu gaismu.
5. Saslēgt shēmu, kas redzama attēlā zemāk.
6. Pieslēgt shēmu sprieguma avotam.
7. Nospiest vispirms vienu pogu, pēc tam otru.
8. Novērot, vai pēc pirmās pogas nospiešanas RGB diode spīd ar dzeltenu gaismu, bet pēc otrās pogas nospiešanas RGB diode spīd ar gaiši zilu gaismu.
9. Nospiežot abas pogas reizē, degs balta gaisma.

RGB diodes vadība bez diodēm



RGB diodes vadība ar diodēm



4.4. DARBA LAPA. RGB mirdzdiodes vadība ar programmu

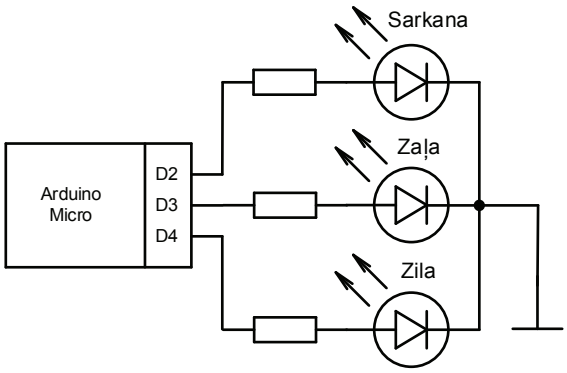
Mērķis

Apgūt RGB diožu vadīšanu ar programmas palīdzību.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors (330 Ω)	3 gab.
RGB mirdzdiode	1 gab.
Montāžas vads	Vairāki

Izveidojamā shēma



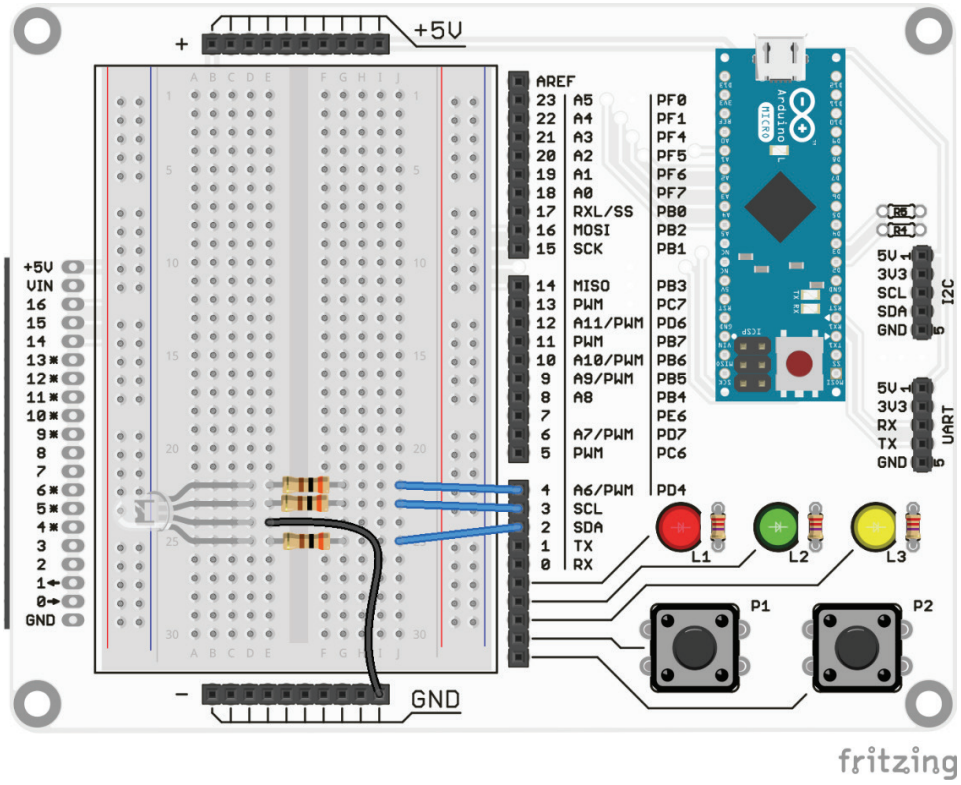
Shēma, lai vadītu RGB diodi ar programmas palīdzību.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā.
2. Uzrakstīt doto vadības programmu.
3. Pieslēgt shēmu sprieguma avotam.
4. Ielādējiet programmu Arduino atmiņā.
5. Novērojiet, kā ik pēc pussekundes mainās diodes krāsa.

```
void setup ()
{
   pinMode(4,OUTPUT);
      //Uzstādām 4. MK kāju kā izeju
   #define greenLEDON digitalWrite(4,HIGH)
      //Definējam, kā ieslēgt zaļo diodi
   #define greenLEDOFF digitalWrite(4,LOW)
      //Definējam, kā izslēgt zaļo diodi
   pinMode(3,OUTPUT);
   #define blueLEDON digitalWrite(3,HIGH)
   #define blueLEDOFF digitalWrite(3,LOW)
   pinMode(2,OUTPUT);
   #define redLEDON digitalWrite(2,HIGH)
   #define redLEDOFF digitalWrite(2,LOW)
}
void RGBColor (boolean red,boolean green,
                                boolean blue)
   //Izveidojam funkciju RGB diodes
   //darbināšanai
{
   if (red){redLEDON;} else {redLEDOFF;}
     //Ja red == 1, tad ieslēdzam sarkano diodi,
      //citos gadījumos izslēdzam diodi
   if (green){greenLEDON;} else {greenLEDOFF;}
   if (blue){blueLEDON;} else {blueLEDOFF;}
}
void loop ()
{
   //Ieslēdzam 7 krāsas RGB diodei
   RGBColor(1,0,0);
                  //Sarkana
   delay(500);    //Pagaidīt pussekundi
   RGBColor(0,1,0);
                  //Zaļa
   delay(500);
   RGBColor(0,0,1);
                  //Zila
   delay(500);
   RGBColor(1,1,0);
                  //Dzeltena
   delay(500);
   RGBColor(0,1,1);
                  //Gaiši zila
   delay(500);
   RGBColor(1,0,1);
                  //Violeta
   delay(500);
   RGBColor(1,1,1);
                  //Balta
   delay(500);
}
```

RGB diodes vadība ar programmu



4.5. DARBA LAPA. Tranzistora lietošana

Mērķis

Apgūt tranzistoru darbību.

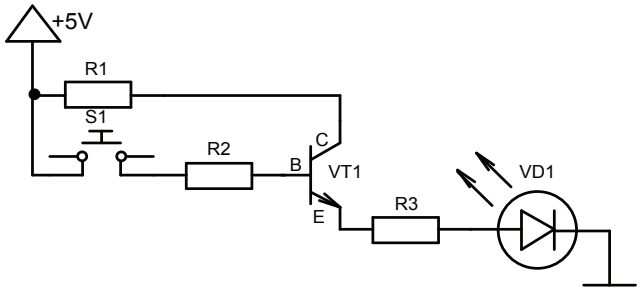
Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā "Tranzistoru ieslēgšana ar pogu ".
2. Pieslēgt shēmu sprieguma avotam.
3. Nospiest pogu.
4. Novērot, vai iedegas mirdzdiode.
5. Saslēgt shēmu, kas redzama attēlā zemāk.
6. Pieslēgt shēmu sprieguma avotam.
7. **Nekādā gadījumā nesavienot brīvos vadus!**
8. Pielikt divus pirkstus, katru pie sava vada.
9. Novērot, vai diode iemirdzas vāji.

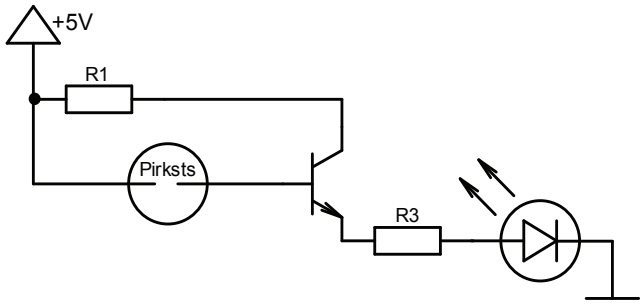
Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Poga	1 gab.
Rezistors R3 (330 Ω)	1 gab.
Rezistors R2 (10 kΩ)	1 gab.
Rezistors R1 (220 Ω)	1 gab.
Mirdzdiode	1 gab.
NPN tranzistors BC517	1 gab.
Montāžas vads	Vairāki

Izveidojamās shēmas

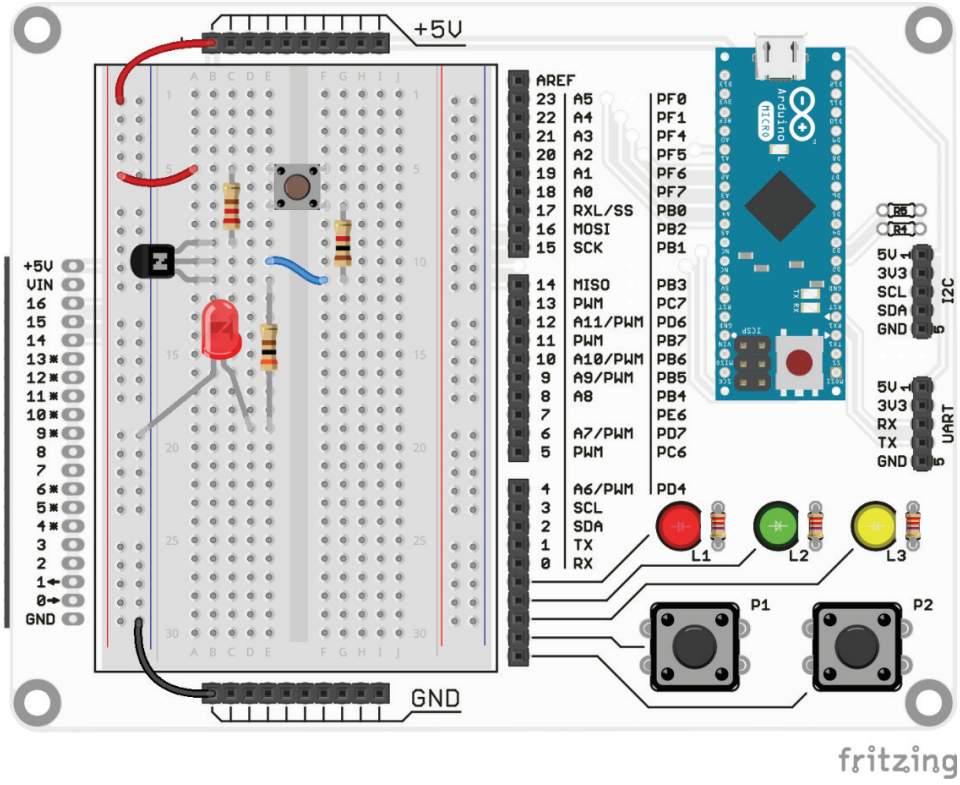


Nospiežot pogu, tiks ieslēgts tranzistors un iedegsies mirdzdiode.

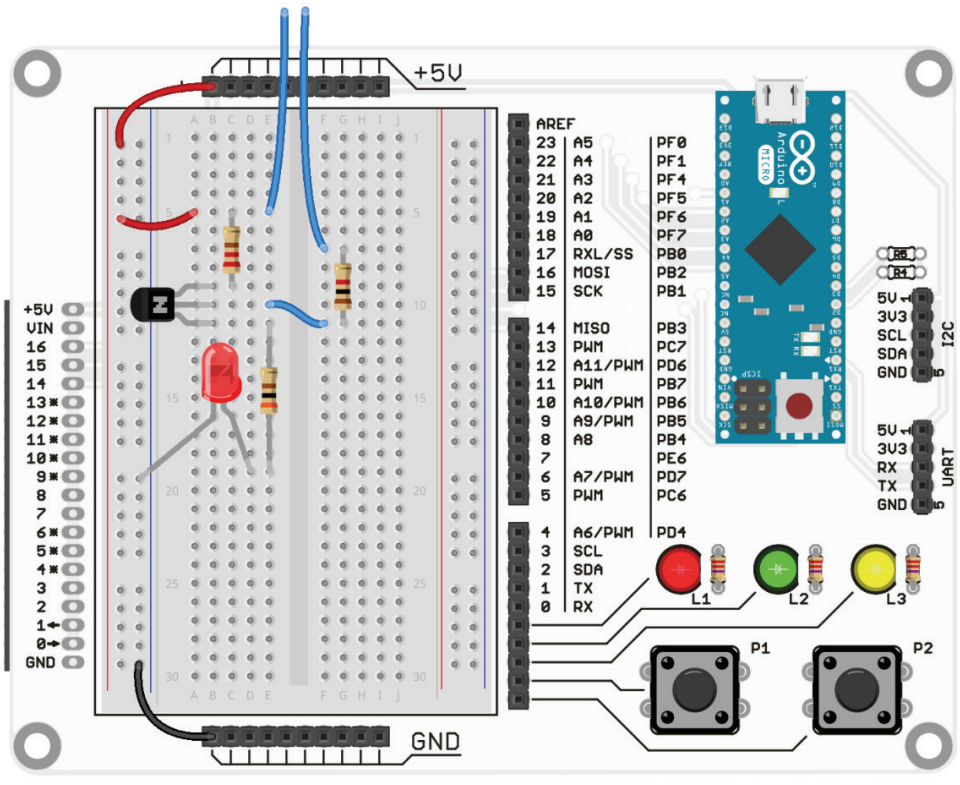


Pieliekot pirkstu norādītajā vietā, vāji iedegsies mirdzdiode.

Tranzistoru ieslēgšana ar pogu



Tranzistoru ieslēgšana ar pirkstu



4.6. DARBA LAPA. Multivibrators

Mērķis

Apgūt dažādu komponentu mijiedarbību.

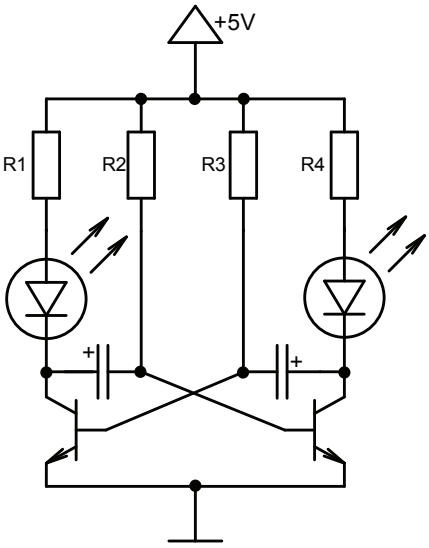
Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors R1, R4 (330 Ω)	2 gab.
Rezistors R1, R3 (10 kΩ)	2 gab.
NPN tranzistors BC517	2 gab.
Mirdzdiode	2 gab.
Kondensators (100 µF)	2 gab.
Montāžas vads	Vairāki

Darba izpildes soļi

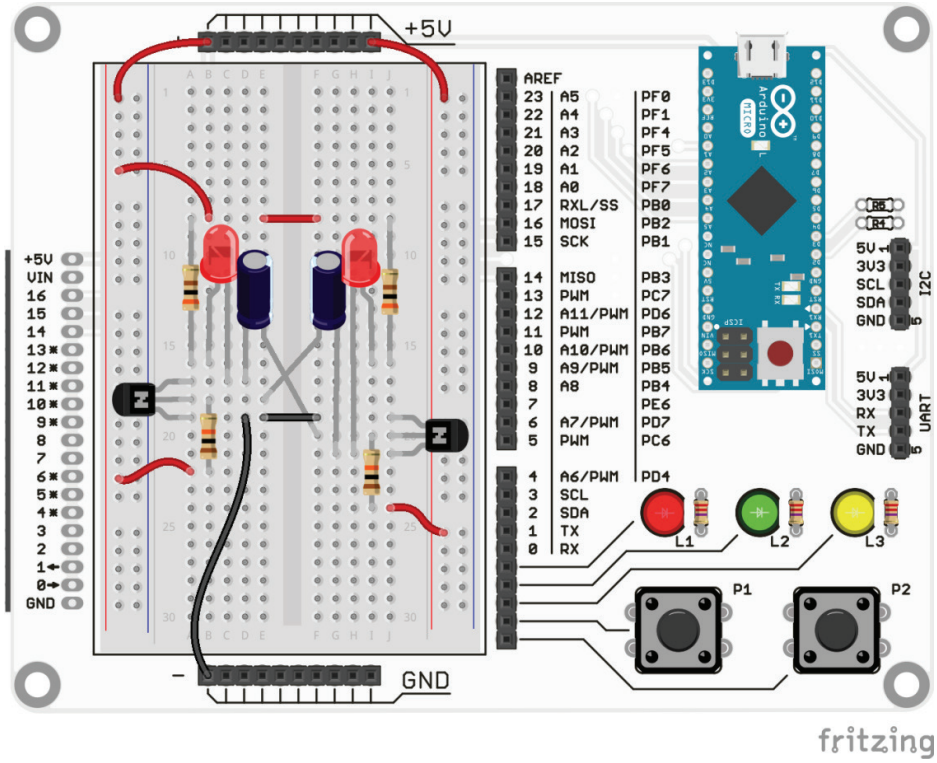
1. Saslēgt shēmu, kas redzama attēlā "Multivibratora shēma".
2. Pieslēgt shēmu sprieguma avotam.
3. Novērot, vai diodes mirkšķinās pamīšus.
4. Nomainot rezistorus R2, R3 vai kondensatorus, mainīsies mirkšķināšanas ātrums.

Izveidojamā shēma



Pareizi saslēdzot shēmu, iegūsiet pamīšus mirgojošas gaismiņas.

Multivibratora shēma



4.7. DARBA LAPA. Multivibrators ar programmu

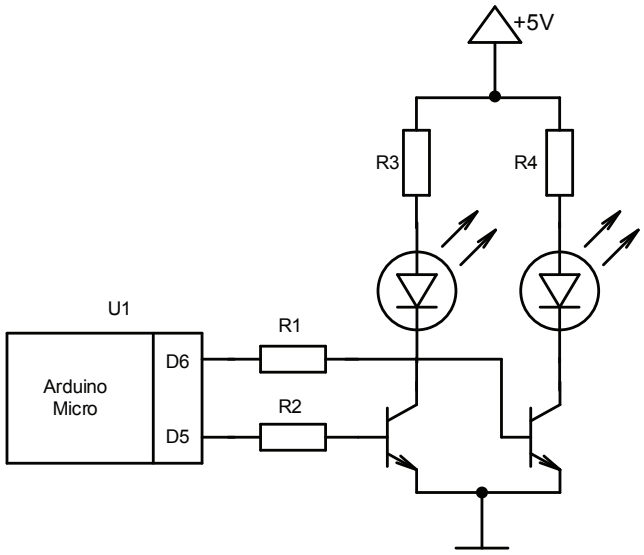
Mērķis

Apgūt tranzistora vadību ar programmu.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors R3, R4, (330 Ω)	2 gab.
Rezistors R1 un R2 (1 kΩ)	2 gab.
NPN tranzistors BC517C	2 gab.
Mirdzdiode	2 gab.
Montāžas vads	Vairāki

Izveidojamā shēma



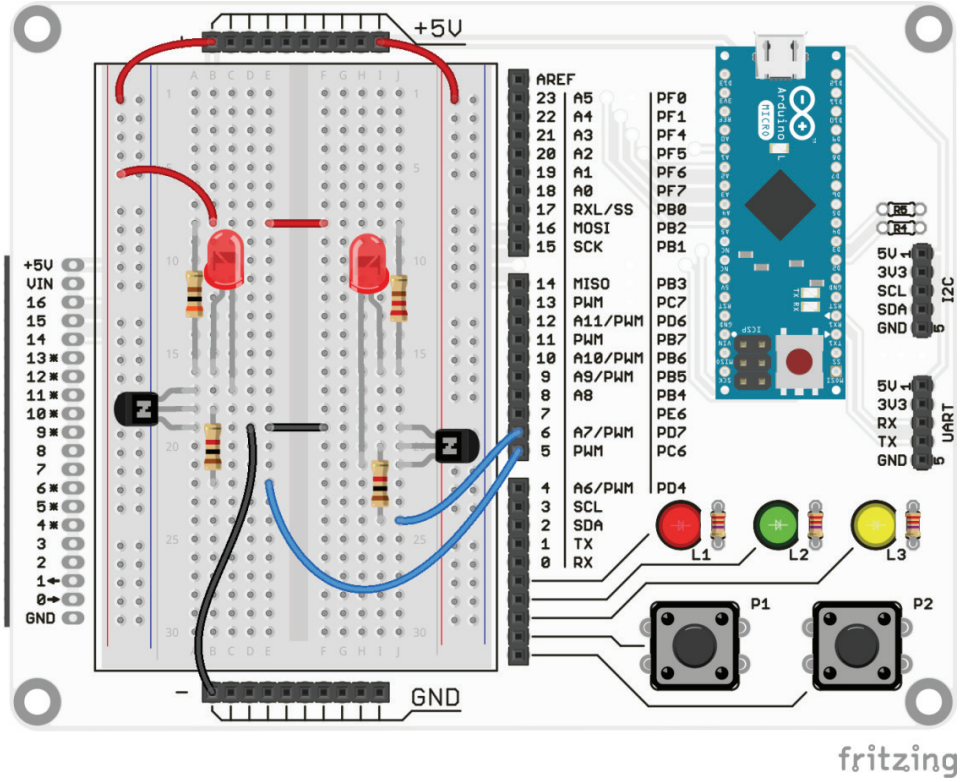
Shēma, kā izveidot multivibratoru ar programmu.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā "Multivibrators ar programmu ".
2. Pieslēgt shēmu sprieguma avotam. Uzrakstīt doto vadības programmu.
3. Ielādēt programmu Arduino un novērot gaismiņas, kas mainās pamīšus.

```
void setup ()
{
   pinMode(5,OUTPUT);
      //Uzstādām 5. MK kāju kā izeju
   #define led1ON digitalWrite(5,HIGH)
      //Definējam, kā ieslēgt 1. diodi
      //Izveidojam definīciju 1. diodes
      //ieslēgšanai
   #define led1OFF digitalWrite(5,LOW)
      //Definējam, kā ieslēgt 2. diodi
      //Izveidojam definīciju 2. diodes
      //izslēgšanai
   pinMode(6,OUTPUT);
   #define led2ON digitalWrite(6,HIGH)
   #define led2OFF digitalWrite(6,LOW)
}
void loop ()
{
   led1ON;led2OFF;
      //Ieslēdzam 1. diodi, izslēdzam 2. diodi
   delay(500);   //Pagaidām pussekundi
   led1OFF;led2ON;
      //Izslēdzam 1. diodi, ieslēdzam 2. diodi
   delay(500);   //Pagaidām pussekundi
}
```

Multivibrators ar programmu



5. SENSORI

5.1. Mērķis

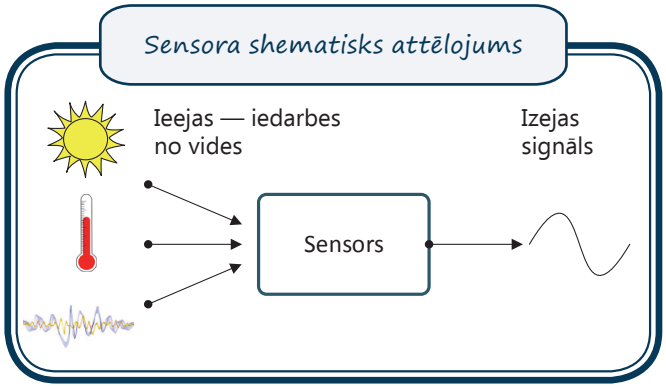
Temata mērķis ir sniegt vispārīgas zināšanas par sensoriem, to veidiem, būtiskāko sensoru tipu darbības principiem un potenciālu izmantojumu robotikā.

5.2. Teorētiskā daļa

5.2.1. Ievads

Sensors ir elements, kas kādu ārēju fizikālu iedarbību spēj pārvērst izejas signālā, kuru var izmantot tālākai apstrādei, vadībai vai lēmumu pieņemšanai. Arī cilvēks izmanto sensorus — acis, ausis un ādu —, lai iegūtu informāciju par apkārtējo pasauli un darbotos atbilstoši saviem mērķiem un vajadzībām.

Atbilstoši kibernētikas aizsācēja Norberta Vīnera (*Norbert Wiener*) priekšstatam — robots ir vadāma sistēma, kas izmanto savu uztveri jeb ieeju un atbilstoši tai maina savu uzvedību tādā veidā, lai sasniegtu noteiktu mērķi. Tādējādi robota ieeju veido viens vai vairāki sensori, kuru sniegtā informācija tiek izmantota robota vadībai. Sensoru darbība shematiski redzama attēlā.



Katrai dabas parādībai — temperatūrai, svaram, ātrumam u. c. — parasti ir nepieciešami īpaši pielāgoti sensori, kas attiecīgo vides iedarbību spēj pārvērst elektriskā signālā, kuru tālāk var izmantot mikroprocesori vai citas iekārtas. Atbilstoši sensoru darbības fizikālajai dabai tos var iedalīt vairākās grupās:

Iedarbības tips	Mērījums — kvantitāte, ko var izmērīt
Optiska — gaismas iedarbība	Gaismas intensitātes samazinājums — cik lielu daudzumu no izstarotās gaismas apstarotais objekts absorbē (patur sevī un neatstaro). Šī tipa sensori ir gan digitālās fotokameras, gan attāluma mērītāji u. c. līdzīgi sensori.
Mehāniska iedarbība	Šī tipa sensori ir guvuši ļoti plašu izplatību un tiek izmantoti svara, spiediena, ātruma, paātrinājuma, pozīcijas un stiepes spēka mērīšanai. Jāpatur prātā, ka pat parasta poga var būt sensors, ar kuru var noteikt, piemēram, sadursmi ar šķērsli.
Elektriska iedarbība	Šī sensoru grupa var sniegt informāciju par lādiņa lielumu, spriegumu, strāvu, maiņstrāvas fāzi, polarizāciju un elektrovadītspēju.
Magnētiska iedarbība	Šie sensori ļauj noteikt magnētiskā lauka polarizāciju, plūdumu un stiprumu (līdzīgi kompasam).
Termiskā iedarbība jeb siltuma iedarbība	Šie sensori ļauj noteikt temperatūras lielumu, tās izmaiņas, kā arī temperatūras vadāmību, t. i., cik labi konkrētais materiāls vada siltumu.
Skaņa — akustiska iedarbība	Skaņas viļņa īpašības — amplitūda, frekvence, polarizācija, frekvenču spektrs, izplatīšanās ātrums vidē u. c. Vienkāršākiem pielietojumiem pietiek ar skaņas amplitūdu (skaļumu) un frekvenci (skaņas toni).
Bioloģiska un ķīmiska iedarbība	Gāzes vai šķidruma koncentrācija noteiktā vidē. Tiek izmantoti gāzu analizatoros un dažāda veida ķīmisko vielu noplūdes noteicējos.

Lai šie sensori darbotos, tiek izmantotas konkrētas likumsakarības un materiālu īpašības. Te tikai daži no tiem:

1. **Faradeja likums** — vadītāja spole pretojas izmaiņām magnētiskajā laukā, ģenerējot elektrisko strāvu un spriegumu magnētiskajam laukam pretējā virzienā.
2. **Fotovadītspējas efekts** — noteiktam pusvadītājam saskaroties ar gaismas starojumu, tā vadītspēja jeb elektriskā pretestība samazinās, t. i., tas kļūs par labāku vadītāju.

Šīs likumsakarības un to plašais pielietojums ikdienā ir labs iemesls, lai lasītājs gūtu praktisku interesi par fiziku skolā vai augstskolā. Sensori ir atrodami visur — radioaparātos, automašīnās, ledusskapjos, putekļsūcējos, medicīnas iekārtās un daudz

kur citur. Ja zinātne nebūtu atklājusi minētās likumsakarības un inženieri nebūtu atraduši to praktisku pielietojumu, mūsdienās automatizācija nepastāvētu. Tas nozīmē, ka daudz lietas nāktos darīt ar rokām vai salīdzinoši vienkāršiem mehanizācijas paņēmieniem.

5.2.2. Kā izvēlēties sensoru

Parasti sensoru izvēli nosaka praktiski apsvērumi, izmērs, svars, cena. Tomēr specifiskos gadījumos, kad ir svarīgas kādas noteiktas sensora vai pētāmā objekta īpašības, tad par labu izvēlei var kalpot viens vai daži sensora raksturlielumi.

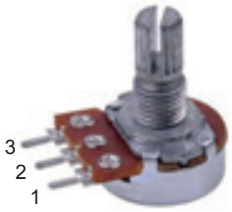
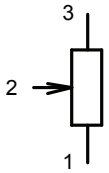
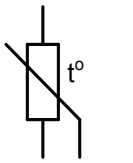
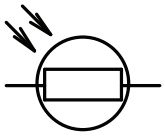
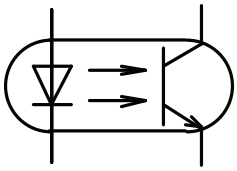
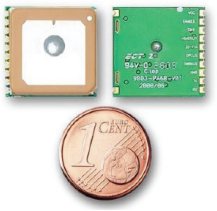
> ***Atcerieties!*** *Nav universālu sensoru, kas ir vienlīdz labi visiem pielietojumiem. Pirms izvēlēties konkrētu sensoru, jābūt skaidram, kas ir noteicošais raksturlielums, un tikai pēc tam var sākt meklēt konkrētu sensoru.*

Būtiskākie raksturlielumi, kas nosaka konkrēta sensora izvēli, ir apkopoti tabulā:

Raksturlielumu grupa	Raksturlielums	Īss skaidrojums
Vides faktori	Darbības temperatūras diapazons, kurā sensors nezaudē savu darbības kvalitāti	Ja darba režīms ir īpašos apstākļos, piemēram, Latvijai dažkārt raksturīgā ziemā ar −30 °C, tad jāizvēlas tādi sensori, kam tehniskajā dokumentācijā attiecīgais rādītājs ir atbilstošs pielietojumam.
	Mitrums	Noturība pret mitrumu ir ļoti būtiska, lai nodrošinātu sensora darbu āra apstākļos.
	Korozijas noturība	Arī korozijas noturība ir saistīta ar pielietojumiem āra apstākļos.
	Izmēri	Dažkārt sensora izmēri ierobežo tā praktisko pielietojamību konkrētā izstrādājumā.
	Pārlieku intensīva iedarbība	Šis rādītājs ir svarīgs, lai, piemēram, temperatūras sensors pēc īslaicīgas temperatūras pārsniegšanas turpinātu darboties.
	Noturība pret mehānisku iedarbību — vibrācijas, kritieni no augstuma	Šis rādītājs parasti ir būtisks mobiliem lietojumiem — mobilos telefonos, automašīnās u. c. vidēs, kas saistītas ar vibrācijām.
	Enerģijas patēriņš	Īpaši nozīmīgs rādītājs iekārtās, kas darbojas uz baterijām, t. i., iekārtām ir ierobežots enerģijas daudzums. Šādos apstākļos parasti tiek ziedota sensora precizitāte un citas īpašības, lai taupītu enerģiju.
	Paštesta iespēja (*self-test*)	Rādītājs ir būtisks autonomās iekārtās, kurām ir jānodrošina pašpietiekama darbība, kad tieša cilvēka iejaukšanās nav iespējama.
Ekonomiskie faktori	Cena	Faktors ir būtisks ierobežotu naudas līdzekļu gadījumā.
	Pieejamība	Šis rādītājs parasti ir būtisks liela apjoma ražošanā, kad ir jāpārliecinās par ražošanas iespējamību iecerētajā apjomā.
	Darba mūžs	Darba mūžs nosaka darbības ilgumu atbilstoši tehniskajai dokumentācijai, t. i., sensora darbībai ir jābūt šajā laikā prognozējamai.
Sensora darbības faktori	Jutīgums	Mērāmā lieluma mazāko izmaiņu vērtība, kuru sensors spēj noteikt.
	Darba diapazons	Mērāmā lieluma intervāls, kuru spēj noteikt sensors.
	Stabilitāte	Noturība pret trokšņiem.
	Atkārtojamība	Mērījumu atšķirība vienādos apstākļos.
	Linearitāte	Šis faktors nosaka mērījuma atkarību no apstākļu izmaiņām, ja tās tiek vienmērīgi mainītas.
	Kļūda	Mērījuma novirze no patiesās vērtības.
	Atbildes laiks	Laiks, kas nepieciešams, lai veiktu kārtējo mērījumu.
	Frekvence	Cik daudz mērījumus iespējams veikt noteiktā laika intervālā.

5.2.3. Sensoru piemēri

Daži populāri sensori, kurus lieto robotu entuziasti. (attēli no www.robotshop.eu)

Sensors	Apzīmējums shēmās	Īss skaidrojums
		**Potenciometrs** parasti ir trīs izvadu ierīce. Tas sastāv no rezistora, kuram pievienots slīdkontakts ar sviru. Grozot potenciometra sviru, mainās pretestība starp vidējo un malējiem izvadiem. Potenciometrus bieži izmanto regulācijai vai kā pagrieziena leņķa sensorus, piemēram, servomotoros.
		**Termistors** ir rezistors, kura pretestība mainās atkarībā no temperatūras. Tas sastāv no speciālas keramikas vai polimēriem un to izmanto, lai precīzi mērītu temperatūru nelielos diapazonos.
		**Fotorezistors** — atkarībā no gaismas intensitātes tas maina pretestību, padarot tajā esošos elektronus daudz mobilākus.
	Nav vienota apzīmējuma	**"Parallax Ping" ultraskaņas attāluma mērītājs** — mēra laiku, kurā pienāk ultraskaņas "atbalss". Tādā veidā, zinot izmantotās ultraskaņas izplatības ātrumu gaisā, iespējams aprēķināt attālumu līdz pirmajam objektam, t. i., tuvākajam šķērslim.
	Nav vienota apzīmējuma	**"Lynxmotion" motora pārvietojuma sensors** — rada vairākus impulsus katras motora rotācijas laukā, t. i., veicot vienu pilnu apgriezienu, uz īpašas plates piestiprināts magnēts vai neliels metāla gabaliņš, rada impulsu, kuru var uztvert mikroprocesors, un aprēķināt reālo motora rotācijas ātrumu.
		**Optopāris** ir ierīce, kura tiek izmantota kā atstarotās gaismas sensors. Optopāris sastāv no infrasarkanās gaismas diodes un fototranzistora. Gaismas diodes izstarotā gaisma, atstarojoties pret kādu virsmu, nonāk fototranzistora bāzē. Ja tranzistora bāzē plūst strāva, tranzistors sāk vadīt strāvu starp kolektoru un emiteru. Atstarotās gaismas daudzums nosaka strāvas lielumu, kas arī ietekmē spriegumu uz tranistora, kas arī tiek mērīts.
	Nav vienota apzīmējuma	**SHARP attāluma sensors** — izmanto infrasarkano gaismu, kas tiek impulsu veidā izstarota, un atstarotais impulss tiek uztverts miniatūrā fotoelementu masīvā. Atkarībā no konkrētā fotoelementa, kas tiek iedarbināts, iespējams aprēķināt objekta attālumu līdz sensoram, pret kuru gaismas impulss atstarojies. attālums tiek kodēts sprieguma veidā intervālā 0–5 V, kas ļauj mikroprocesoram interpretēt mērījumu.
	Nav vienota apzīmējuma	**"MediaTek AGPS" sensors** — daudz sarežģītāks par iepriekšējiem sensoriem, kas sniedza GPS koordinātas. Tas ir iegūstamas, izmantojot īpašu datu apmaiņas protokolu un atbilstošu programmatūru mikrokontrolleri. Šāda tipa sensori ir sastopami mobilajos telefonos un citās pārnēsājamās iekārtās.

Sensoru daudzveidība mūsdienās ir tik milzīga, ka šajā mācību materiālā tos nav iespējams iekļaut. Taču lasītājs noteikti var apmeklēt kādu no robotikas entuziastiem veltītiem interneta veikaliem un gūt pilnīgāku priekšstatu par pieejamajiem sensoriem un to raksturlielumiem.

5.1. DARBA LAPA. Leņķa sensors

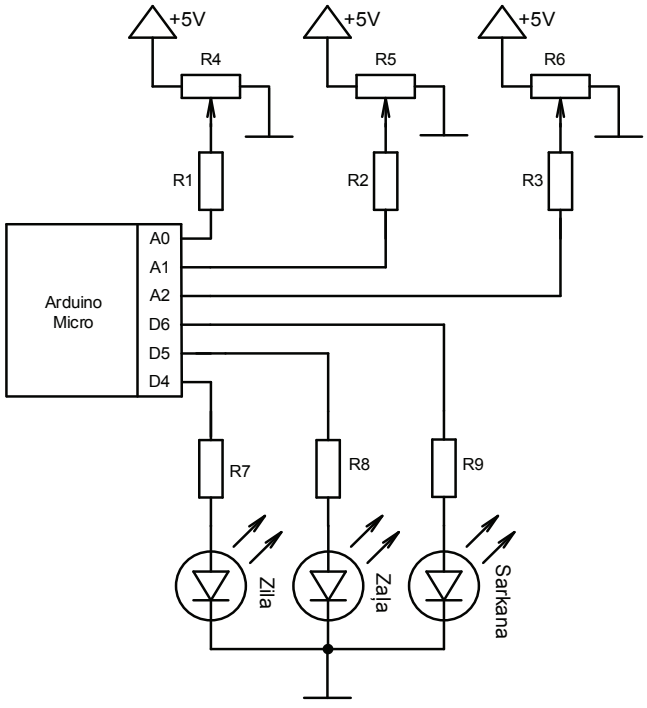
Mērķis

Apgūt potenciometra lietošanu.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistori R1, R2, R3 (1 k Ω)	3 gab.
Potenciometri R4, R5, R6 (50 k Ω)	3 gab.
Rezistori R7, R8, R9 (330 Ω)	3 gab.
RGB mirdzdiode	1 gab.
Montāžas vads	Vairāki

Izveidojamā shēma



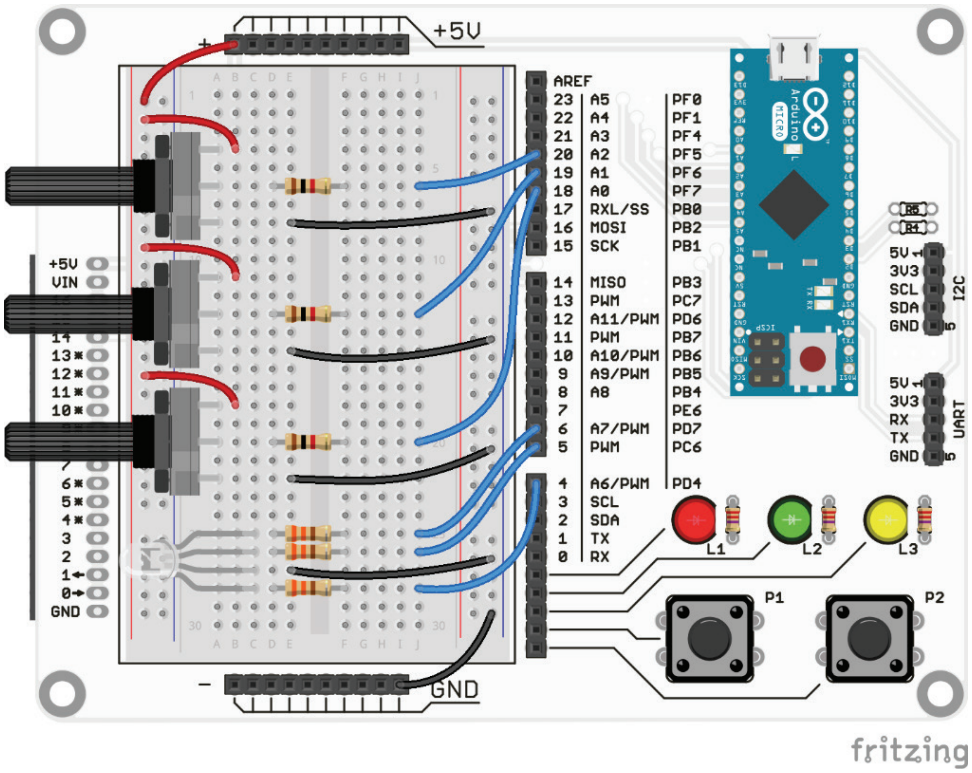
RGB diodes krāsas mainīšana ar trīs potenciometriem.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā.
2. Pieslēgt shēmu sprieguma avotam.
3. Ielādēt doto programmu.
4. Pagriezt potenciometra sviras.
5. Novērot, kā mainās RGB diodes krāsa.

```
int greenPin = 6;    //Zaļās diodes pins
int bluePin = 5;     //Zilās diodes pins
int redPin = 4;      //Sarkanās diodes pins
int pot1Pin = A0;    //Pirmais potenciometrs
int pot2Pin = A1;    //Otrais potenciometrs
int pot3Pin = A2;    //Trešais potenciometrs

void setup ()
{
   #define redRead analogRead(pot3Pin)/4
      //Nolasa potenciometra vērtību un izdala
      //ar 4
   #define greenRead analogRead(pot2Pin)/4
      //Nolasa potenciometra vērtību un izdala
      //ar 4
   #define blueRead analogRead(pot1Pin)/4
      //Nolasa potenciometra vērtību un izdala
   ...//ar 4
}
void RGBLED (int red,int green,int blue)
   //Funkcija RGB diodes vadīšanai
{
   analogWrite(redPin,red);
      //Norāda sarkanās krāsas intensitāti
   analogWrite(greenPin,green);
      //Norāda zaļās krāsas intensitāti
   analogWrite(bluePin,blue);
      //Norāda zilās krāsas intensitāti
}
void loop ()
{
   RGBLED(redRead,greenRead,blueRead);
      //Iedarbina RGB LED krāsu uzstādīšanas
   ...//funkciju
}
```

RGB mirdzdiodes krāsu mainīšana



5.2. DARBA LAPA. Temperatūras sensors

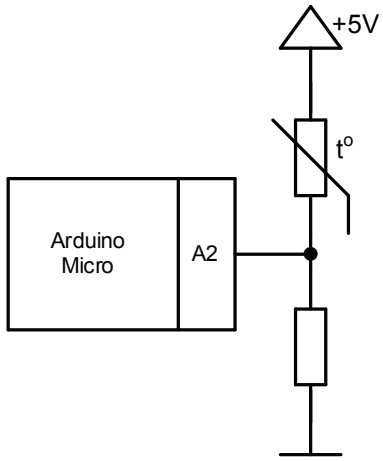
Mērķis

Apgūt termorezistora lietošanu un to, kā informāciju parādīt monitorā.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Termistors	1 gab.
Rezistors (1 k Ω)	1 gab.
Vads	5 gab.

Izveidojamā shēma



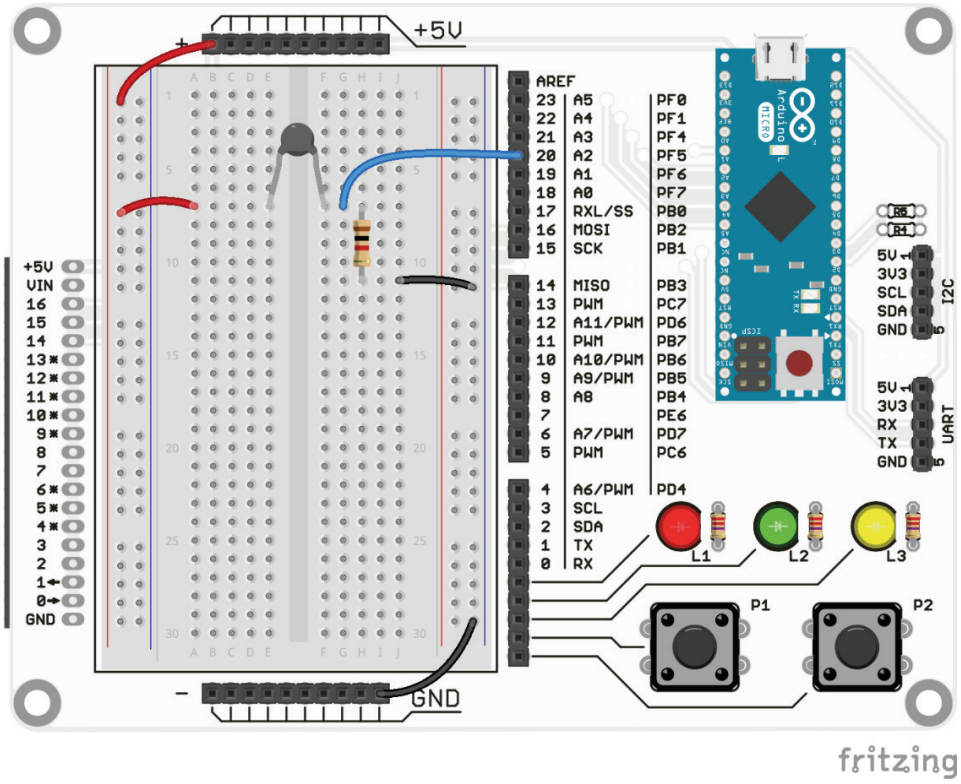
Termorezistora pieslēgšanas shēma temperatūras noteikšanai.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā.
2. Pieslēgt shēmu sprieguma avotam.
3. Ielādēt doto programmu.
4. Atvērt seriālā porta monitoru Rīki → Seriālā porta monitors.
5. Novērot seriālajā monitorā izvadīto temperatūru.
6. Saspiest sensoru pirkstos un novērot temperatūras izmaiņas.

```
int termoPin = A2;   //Termorezistora pins
float tempcoef = 0.07;
   //Sprieguma izmaiņas koeficients rezistoram

void setup ()
{
   #define termoReading analogRead(termoPin)
      //Termorezistora vērtības nolase
   Serial.begin(9600);
      //Tiek ieslēgta komunikācija ar datoru
}
void loop ()
{
   float temperature = termoReading*tempcoef;
      //Temperatūras aprēķināšana
   Serial.print ("Temperatura ir: ");
   Serial.println(temperature);
      //Izvada informāciju uz seriālā porta
      //monitora
   delay(20);
}
```

Termorezistora pieslēgšanas shēma



5.3. DARBA LAPA. Gaismas sensors

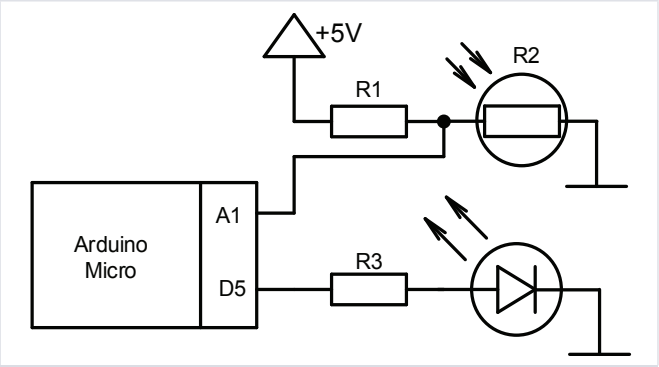
Mērķis

Apgūt fotorezistora lietošanu.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors R1 (1 kΩ)	1 gab.
Fotorezistors R2 (20 kΩ)	1 gab.
Rezistors R3 (330 Ω)	1 gab.
Mirdzdiode	1 gab
Vads	Vairāki

Izveidojamā shēma



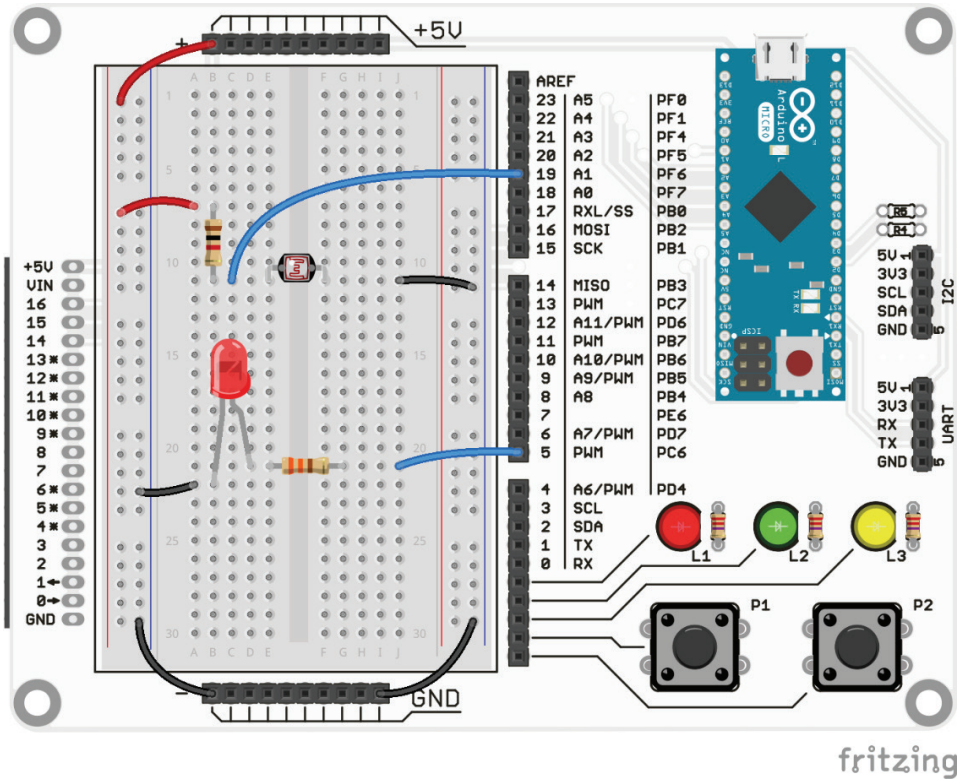
Fototranzistora pieslēgšanas shēma ar apgaismojuma diodi. Diode ieslēgsies, kad satumsīs.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā.
2. Pieslēgt shēmu sprieguma avotam.
3. Ielādēt doto programmu.
4. Atvērt seriālā porta monitoru **Rīki → Seriālā porta monitors.**
5. Novērot seriālajā monitorā izvadīto gaismas līmeni.
6. Aizklāt fotorezistoru ar roku.
7. Novērot, vai iedegas mirdzdiode.
8. Ja mirdzdiode neiedegas, mainīt "darktreshold" vērtību uz zemāku, nekā parāda seriālā porta monitors, aizklājot diodi.

```
int fotoPin = A1;   //Fotorezistora pins
int ledPin = 5;     //Gaismas diodes pins
int darktreshold = 900;   //Tumsas robeža

void setup ()
{
   pinMode(ledPin,OUTPUT);
      //Uzstāda LED kāju kā izeju
   #define fotoReading analogRead(fotoPin)
      //Fotorezistora vērtības nolase
   #define ledON digitalWrite(ledPin,HIGH)
      //Definē LED izslēgšanas darbību
   #define ledOFF digitalWrite(ledPin,LOW)
      //Definē LED ieslēgšanas darbību
   Serial.begin(9600);
      //Ieslēdz komunikāciju ar datoru
}
void loop ()
{
   Serial.print ("Gaismas vertiba ir: ");
   Serial.println(fotoReading);
      //Izvada informāciju uz seriālā porta
   ....//monitora
   if (fotoReading > darktreshold)
      //Patiess, ja fotonolase lielāka
   ....//par tumsas robežu
   {
      ledON;  //Ja paliek tumšs, ieslēdz diodi
   }
   else
   {
      ledOFF;
         //Ja paliek gaišs, izslēdz diodi
   }
}
```

Fotorezistora pieslēgšana



5.4. DARBA LAPA. Šķēršļu un gaismas atstarošanās sensors

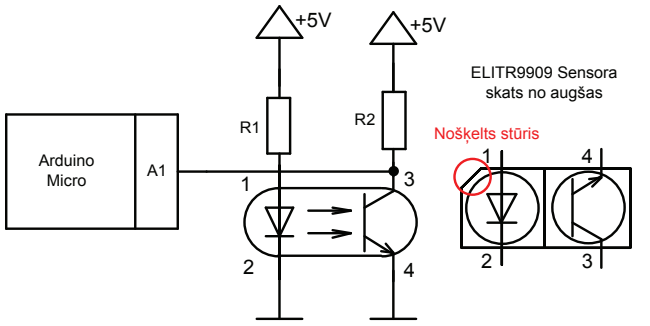
Mērķis

Apgūt optopāra lietošanu.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors R1 (330 Ω)	1 gab.
Rezistors R2 (10 kΩ)	1 gab.
Optopāris ELITR9909	1 gab.
Vads	Vairāki

Izveidojamā shēma



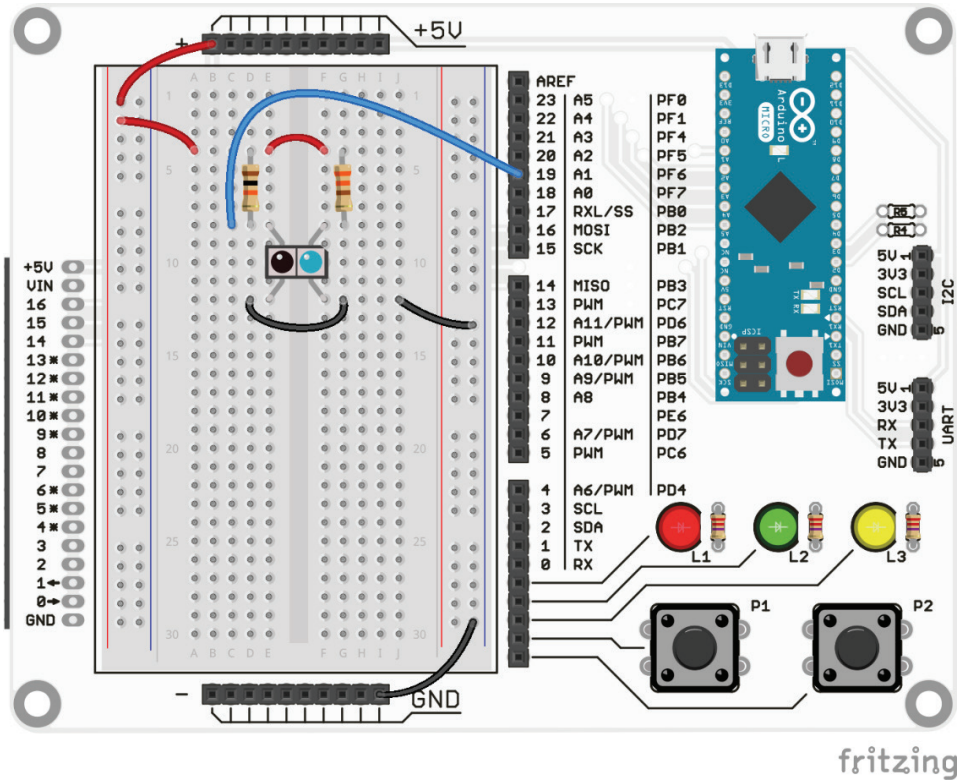
Optopāra pieslēgšana un sensora datu nolasīšana.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā.
2. **Uzmanību — pievērst uzmanību pareizai sensora pieslēgšanai un MK kāju izvietojumam!**
3. Pieslēgt shēmu sprieguma avotam.
4. Ielādēt doto programmu.
5. Atvērt seriālā porta monitoru Rīki → Seriālā porta monitors.
6. Novērot informāciju seriālā porta monitorā.
7. Pielikt 4 mm attālumā no sensora baltu lapu.
8. Novērot sensora vērtību.
9. Pielikt 4 mm attālumā melnu lapu.
10. Novērot sensora vērtību.
11. Dažādos attālumos pielikt sensoram priekšā lapu.
12. Novērot rezultātus.

```
int optoPin = A1;   //Optopāra sensora pins
int objecttreshold = 1000;   //Objekta robeža
int whitetreshold = 150;
   //Baltās krāsas robeža
void setup ()
{
   #define optoReading analogRead(optoPin)
      //Optopāra signāla nolase
   Serial.begin(9600);
      //Ieslēdz komunikāciju ar datoru
}
void loop ()
{
   Serial.print ("Sensora radijums ir: ");
   Serial.println(optoReading);
      //Parāda sensora rādījumu
   if (optoReading < objecttreshold)
      //Patiess, ja rādījums mazāks
      //par objekta robežu
   {
      Serial.println ("Sensoram prieksa ir
                                    Objekts!");
      if (optoReading < whitetreshold)
         //Patiess, ja rādījums mazāks
         //par gaišo robežu
      {
         Serial.println ("Objektam ir gaisa
                                    krasa!");
      }
      else
         //Ja rādījums ir lielāks par gaišo
         //robežu
      {
         Serial.println ("Objektam ir tumsa
                                    krasa");
      }
   }
   else
      // Ja rādījums ir lielāks par objekta
      //robežu
   {
      Serial.println ("Sensoram prieksa nav
                                    Objekts!");
   }
   delay(500);
}
```

Optopāra pieslēgšanas shēma



5.5. DARBA LAPA. Kapacitīvais sensors

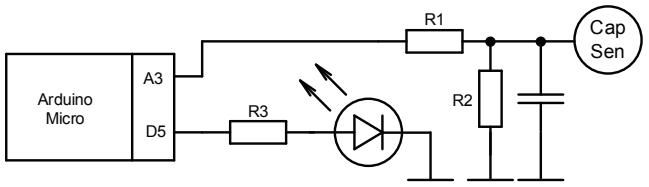
Mērķis

Izveidot vienkāršu kapacitīvo sensoru.

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors R1 (4,7 kΩ)	1 gab.
Rezistors R2 (47 kΩ)	1 gab.
Rezistors R3 (330 Ω)	1 gab.
Kondensators (15 pF)	1 gab.
Mirdzdiode	1 gab.
Kapacitīvā poga no alumīnija vai vara follija	1 gab.
Vads	Vairāki

Izveidojamā shēma



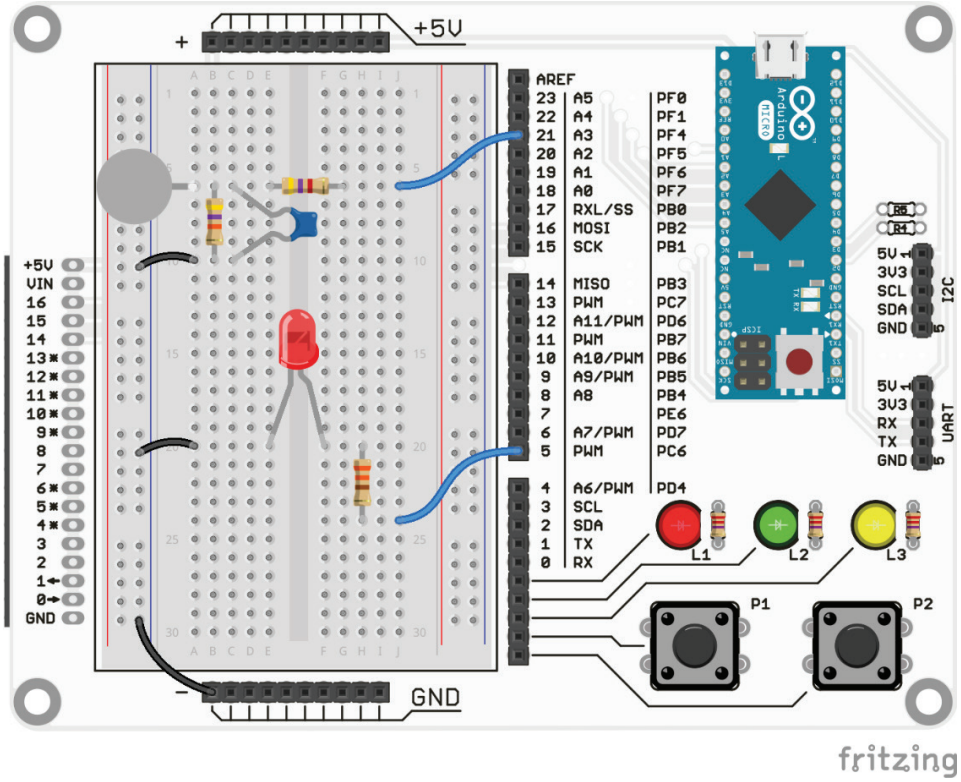
Kapacitīvās pogas shēma. Poga ir no alumīnija vai vara folija, ar vadu tā pievienota pie shēmas.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā.
2. Pieslēgt shēmu sprieguma avotam.
3. Pievienot vara vai alumīnija plāksni.
4. Ielādēt doto programmu.
5. Ar pirkstu pieskarties alumīnija plāksnei.
6. Novērot, vai diode iedegas.
7. Atvērt seriālā porta monitoru **Rīki → Seriālā porta monitors.**
8. Novērot seriālā porta monitorā sensora nolases vērtības atkarībā no pieskaršanās.

```
int capPin = A3; //Kapacitīvā sensora pins
int ledPin = 5;  //Diodes pins

void setup ()
{
   #define capReading analogRead(capPin)
      //Kapacitīvā sensora nolase
   pinMode(ledPin,OUTPUT);
      //Uzstāda LED kāju kā izeju
   #define ledON digitalWrite(ledPin,HIGH)
      //Ieslēdz gaismas diodi
   #define ledOFF digitalWrite(ledPin,LOW)
      //Izslēdz gaismas diodi
   Serial.begin(9600);
      //Ieslēdz komunikāciju ar datoru
}
int CapacitivSen ()
   //Funkcija kapacitīvā sensora nolasei
{
   pinMode(capPin,OUTPUT);
      //Uzstāda nolases MK kāju kā izeju
   digitalWrite(capPin,HIGH);
      //Uzlādē sensoru
   pinMode(capPin,INPUT);
      //Uzstāda nolases MK kāju kā ieeju
   int cap = capReading;
      //Nolasa sensora vērtību
   return cap;
      //Funkcija atgriež nolasīto vērtību
}
void loop ()
{
   Serial.println(CapacitivSen());
      //Nolasa vērtību uz ekrāna
   if (CapacitivSen()>2){
      //Ieslēdz diodi, ja nolasītā vērtība
      //ir lielāka par 2
   ledON;
}
   else {
      ledOFF;
   }
   delay(10);
}
```

Kapacitatīvā sensora shēma



6. MOTORI UN TO VADĪBA

6.1. Mērķis

Temata mērķis ir iepazīstināt ar dažādu motoru veidiem un to vadības pamatiem. Šī nodaļa nesniedz izsmeļošu aprakstu par visiem elektrodzinēju veidiem, bet tikai tiem, kas pieejami konstruktorā, t. i., patstāvīgā magnēta līdzstrāvas dzinēju un servomotoru.

6.2. Teorētiskā daļa

6.2.1. Ievads

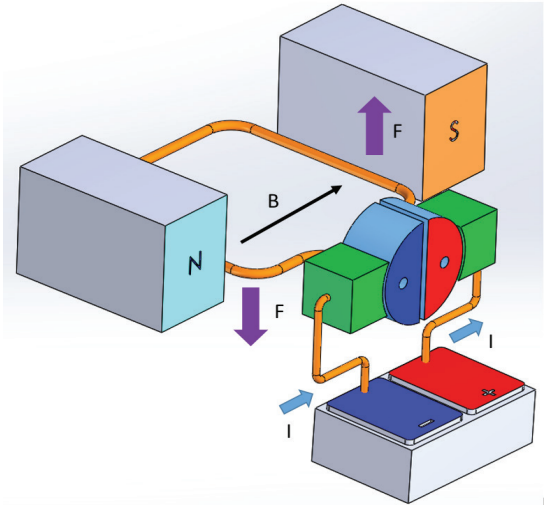
Elektrodzinējs ir elektrotehniska iekārta, kas ļauj elektrisko enerģiju pārvērst mehāniskā enerģijā, t. i., motors tiek griezts, jo motora tinumos plūst elektroenerģija. Elektromotori laika gaitā ir piedzīvojuši dažādus tehniskos risinājumus, no kuriem visvienkāršākais ir patstāvīgā magnēta līdzstrāvas dzinējs. Tā uzbūve un darbības vienkāršota shēma redzama attēlā.

Neiedziļinoties sīkumos, ir jāzina, ka motora griešanās spēks ***F*** ir tieši proporcionāls strāvai ***I*** un magnētiskā lauka stiprumam ***B***. Tādējādi — jo spēcīgāki magnēti un lielāka strāva, jo lielāks motora radītais spēks. Pēdējos 20 gados, parādoties īpaši spēcīgajiem neodīma magnētiem, tirgū ir pieejami nelieli, bet ļoti spēcīgi elektromotori. Tomēr, runājot par robotikai raksturīgiem pielietojumiem, būtisks ir jautājums par šādu dzinēju vadīšanu, t. i., kā robota vadības mehānismi varētu iedarboties uz dzinēju, lai mainītu tā rotācijas virzienu un ātrumu atbilstoši nepieciešamajam.

6.2.2. Elektromotora ātruma maiņa

Šajā apakšnodaļā tiks aplūkots viens konkrēts veids patstāvīgā magnēta līdzstrāvas dzinēja griešanās ātruma vadībai, kas balstīts uz sprieguma impulsa platuma noteikšanu. Šī tehnika dažkārt tiek izmantota arī analogu signālu iegūšanai ar ciparu tehnikas paņēmieniem. Šīs tehnikas būtība ir ieslēgt un izslēgt strāvu, veidojot impulsu ar noteiktu platumu (laiks, kurā strāvā ir ieslēgta). Mainot "ieslēgtas" strāvas proporciju laikā attiecībā pret "izslēgtu" strāvu, mainās dzinēja griešanās ātrums. Mainot šo proporciju, tiek veikta signāla modulācija (skat. attēlu). Tā arī radies tehnikas nosaukums — impulsa platuma modulācija.

Patstāvīgā magnēta līdzstrāvas dzinēja uzbūves un darbības shēma



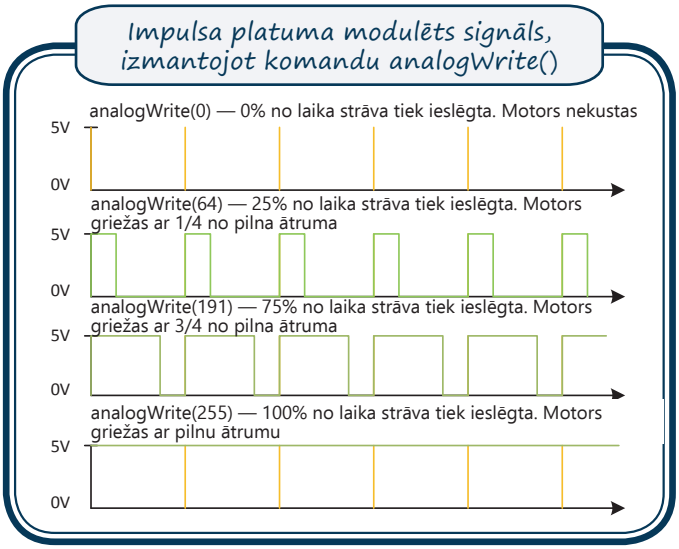
I — strāvas plūšanas virziens: no akumulatora pozitīvā uz negatīvo polu.

B — patstāvīgo magnētu radītā magnētiskā lauka virziens.

F — strāvas plūšanas un magnētiskā lauka radītā spēka virziens.

Kā redzams attēlā, strāva un magnētu magnētiskais lauks rada spēku, kas motora tinumu — rotoru — griež noteiktā virzienā. Tomēr šādi virzīts spēks saglabājas tikai līdz noteiktam brīdim, kad rotors tiks pagriezts par 180 grādiem. Tādēļ, lai nodrošinātu motora nepārtrauktu kustību vēlamajā virzienā, parasti ir vairāki tinumi, kuri pēc kārtas ar kontaktoru (ar zaļu krāsu iezīmēti attēlā) palīdzību pārslēdzas tā, lai tiktu saglabāts pēc iespējas lielāks spēks ***F***.

Mainot pieslēguma polus vietām, spēks mainīs savu virzienu, un motors tiks griezts pretējā virzienā.



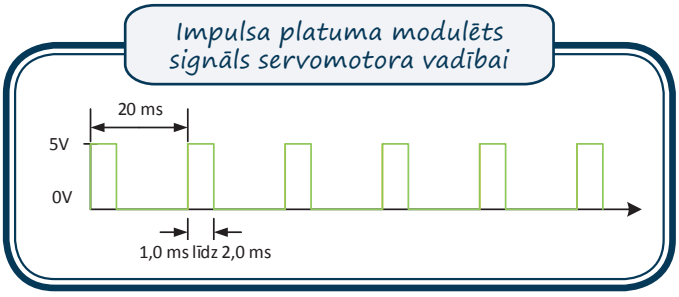
Šajā attēlā oranžās līnija norāda uz laika intervāliem, kas ir atkarīgs no konkrētā mikroprocesora. Arduino tie ir 500 Hz. Zaļā līnija norāda uz laiku, kurā strāva tiek ieslēgta vai izslēgta — impulsa platumu.

Tādējādi, izmantojot komandu **analogWrite()**, programmā iespējams mainīt motoru griešanās ātrumu vai mirdzdiodes mirdzēšanas intensitāti, kā tika demonstrēts iepriekšējās nodaļās.

7.2.3. Servomotora vadība

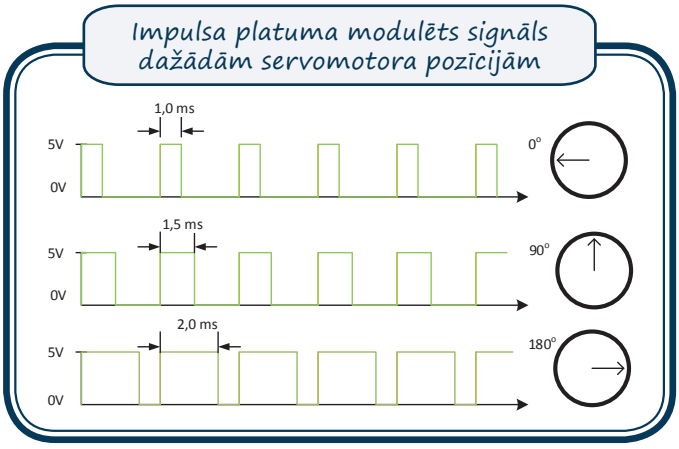
Atšķirībā no vienkārša līdzstrāvas dzinēja, kas aprakstīts iepriekš, servomotors ir īpaša vadības ķēde, kas ļauj ērti vadīt dzinēja ātrumu vai konkrētu pozīciju, kurā dzinēja asij ir jānonāk. Dzinēja vadība tiek realizēta ar trim pieslēgumiem — strāvas pozitīvo (parasti sarkans) un negatīvo pieslēgumu (brūns vai melns), kā arī vadības pieslēgumu (oranžs vai dzeltens).

Vadībai tiek izmantots jau aplūkotais impulsa platuma paņēmienu, bet ar savādāku cikla garumu (skat. attēlu).



Kā redzams, servomotora impulsa cikla garums ir 20 milisekundes, bet impulsa garums ir no 1 līdz 2 milisekundēm. Šie signāla raksturlielumi ir spēkā lielākajai daļai entuziastu līmeņa servomotoru, bet tas noteikti ir jāpārbauda katram konkrētam motora modelim ražotāja specifikācijā.

Servomotora vadības ķēde sagaida impulsu katras 20 milisekundes, bet impulsa platums norāda uz pozīciju, kurā servomotoram ir jānonāk. Piemēram, 1 ms (milisekunde) norāda u 0 grādu pozīciju, bet 2 ms uz 180 grādu pozīciju attiecībā pret noteikto sākuma punktu. Nonākot attiecīgajā pozīcijā, servomotors notur šo pozīciju, t. i., pretojas tās izmaiņām, ja ar kādu ārēju spēku cenšas šo pozīciju mainīt. Grafiski tas attēlots attēlā.



Tāpat kā citiem motoriem, arī servomotoriem ir dažādi parametri, no kuriem būtiskākais ir nostrādāšanas laiks, t. i., laiks, kas nepieciešams, lai veiktu pārvietojumu noteiktā leņķī. Labākajiem entuziastu līmeņu servomotoriem tas ir ap 0,09 sekundēm, lai veiktu pagriezienu par 60 grādiem.

Atkarībā no pielietojuma servomotorus var iedalīt šādās grupās:

- **Pozīcijas servomotors** — visplašāk izplatītais servomotoru veids, kas aprakstīts iepriekš, un ar vadības signāla palīdzību ļauj noteikt motora ass pozīciju attiecībā pret sākuma punktu.
- **Nepārtrauktas rotācijas servomotors** — šī tipa motori ar vadības signālu ļauj noteikt rotācijas virzienu un rotācijas ātrumu. Ja pozīcija zem 90 grādiem, tas griežas vienā virzienā, bet, ja virs, tad — pretējā virzienā. Ātrumu nosaka ar attālumu no 90 grādiem, t. i., 0 vai 180 grādi ļaus griezties ar maksimālu ātrumu attiecīgajā virzienā, bet, 91 vai 89 grādi ar minimālu.
- **Lineāri servomotori** — ar papildu pārnesumu palīdzību tie ļauj pārvietoties uz priekšu un atpakaļ, tie nerotē.

Diemžēl Arduino neļauj vadīt tikpat vienkārši, kā aplūkoto līdzstrāvas dzinēju. Tamdēļ ir izveidota īpaša servomotoru vadības bibliotēka.

7.2.4. *H* tilts

H tilts (tilts) savu nosaukumu ir ieguvis tā līdzības dēļ lielajam "H" burtam, kura visos stūros ir slēdži, bet vidū elektromotors. Parasti tiltu izmanto patstāvīgā magnēta līdzstrāvas dzinēju, elektromagnētu

u. c. līdzīgu elementu darbināšanai, jo ļauj ar mazu strāvu darbināt ievērojami lielākas strāvas elektroiekārtas.

Pārslēdzot slēdžus, iespējams mainīt motora rotācijas virzienu. Ir jāņem vērā, ka slēdžu ieslēgšana un izslēgšana ir jāveic pa pāriem. Tas redzams attēlā "Strāvas plūšana H tiltā".

Ja tiek ieslēgti abi negatīvie vai abi pozitīvie slēdži apakšpusē vai augšpusē, tad dzinējs tiek apturēts, nepieļaujot brīvu rotāciju — tas tiek bremzēts. Vadību var apkopot zemāk dotā tabulā.

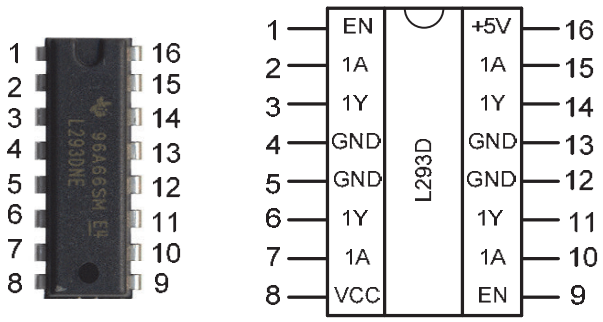
Ja visi slēdži ir izslēgti, tad dzinējs vienkārši ir brīvā kustībā. Ne vienmēr robotikā ar to pietiek, tādēļ dažkārt *H* tilts, lai strauji bremzētu, tiek ieslēgts rotācijai pretējā virzienā, t. i., tiek strauji ieslēgta "atpakaļgaita".

> ***Atcerieties**! Ne viens, ne otrs bremzēšanas paņēmiens nav saudzīgs pret H tiltu vai lietoto strāvas avotu. Tādēļ šāda rīcība nav pieļaujama bez īpaša iemesla, jo var radīt slēdžu vai strāvas avota bojājumus.*

Tiktāl viss ir salīdzinoši vienkārši. Sarežģītāk ir realizēt slēdžus — ja slēdži neder, tad parasti tiek izmantoti releji vai pietiekamas jaudas tranzistori. Releju būtiskākais trūkums ir spēja dzinējus tikai ieslēgt vai izslēgt. Ja nepieciešams regulēt arī rotācijas ātrumu ar iepriekš aplūkoto impulsa platuma modulētu signālu, tad jāizmanto tranzistori. Lielas jaudas nodrošināšanai ir jāizmanto MOSFET tipa tranzistori.

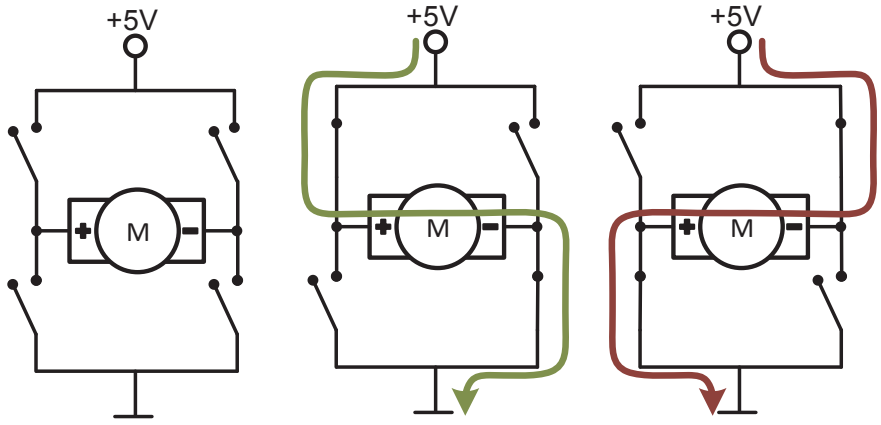
Konkrēts slēgums ir atkarīgs no vairākiem faktoriem, kā arī no izmantoto tranzistoru konkrētiem modeļiem. Mūsdienās, lai nodrošinātu tilta stabilu darbu, tas tiek papildināts ar papildu elementiem. Rūpnieciski ražotiem tiltiem ir viens korpuss, piemēram, tas, kas pieejams konstruktorā — L293D.

L293D mikroshēma un tās apzīmējums shēmā



L293D mikroshēma sastāv no diviem *H* tiltiem un paredzēta divu motoru vadībai. Katrai mikroshēmas kājiņai ir sava funkcija, tāpēc ir svarīgi tās nesajaukt, jo citādi mikroshēma var tikt bojāta. Visām mikroshēmām kājiņas tiek numurētas, numerācija sākas pie atzīmes uz korpusa: iešķēluma, punktiņa, nosķeltas malas u. c. un turpinās pretēji pulksteņa rādītaja virzienam. Veidojot shēmu, ir svarīgi ievērot šos numurus un tos numurus, kas norādīti shēmā. Ja nepieciešams uzzināt vairāk par mikroshēmu, informāciju var atrast mikroshēmas datu lapā. Atgādinām, ka datu lapu var atrast, ierakstot interneta pārlūkprogrammā ierīces numuru, kas atrodams uz korpusa, un pievienojot klāt atslēgvārdu "datasheet".

Strāvas plūšana H tiltā



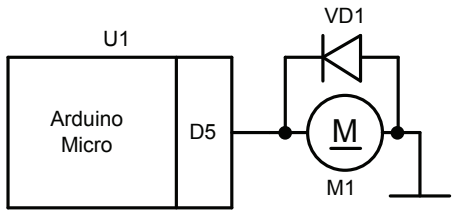
Augšējais kreisais	Augšējais labais	Apakšējais kreisais	Apakšējais labais	Motora darba režīms
Ieslēgts	Izslēgts	Izslēgts	**Ieslēgts**	**Griežas vienā virzienā**
Izslēgts	**Ieslēgts**	**Ieslēgts**	Izslēgts	**Griežas otrā virzienā**
Ieslēgts	**Ieslēgts**	Izslēgts	Izslēgts	Dzinējs tiek bremzēts
Izslēgts	Izslēgts	**Ieslēgts**	**Ieslēgts**	Dzinējs tiek bremzēts

6.1. DARBA LAPA. Motora darbināšana ar programmu

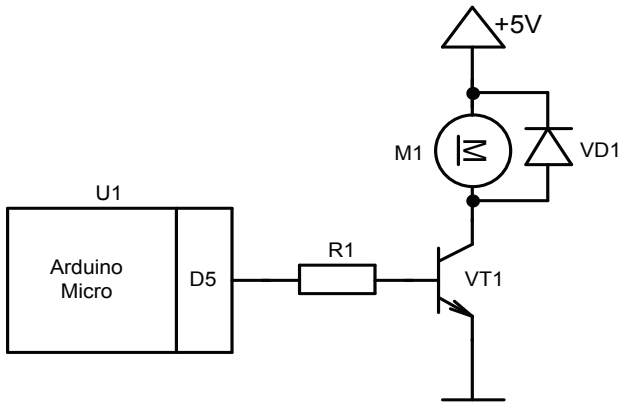
Mērķis

Apgūt motoru vadību ar programmas palīdzību.

Izveidojamās shēmas



Ja motoru pieslēgs pie Arduino bez tranzistora, motors griezīsies ļoti lēni vai negriezīsies vispār.



Motors griezīsies ātrāk, ja to vadīs caur tranzistoru.

Nepieciešamie materiāli

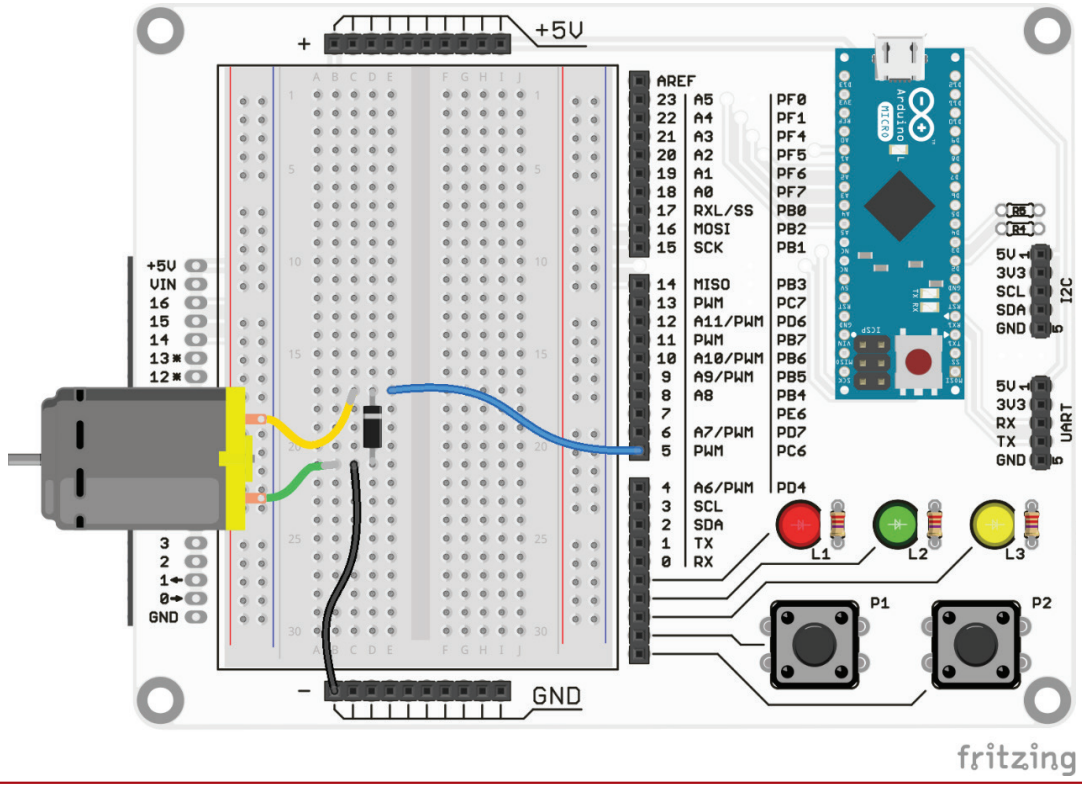
Materiāls/detaļa	Skaits
Rezistors (10 kΩ)	1 gab.
Līdzstrāvas motors (3–6 V)	1 gab.
Diode PH4148	1 gab.
NPN tranzistors BC517	1 gab.
Vads	Vairāki

Darba izpildes soļi

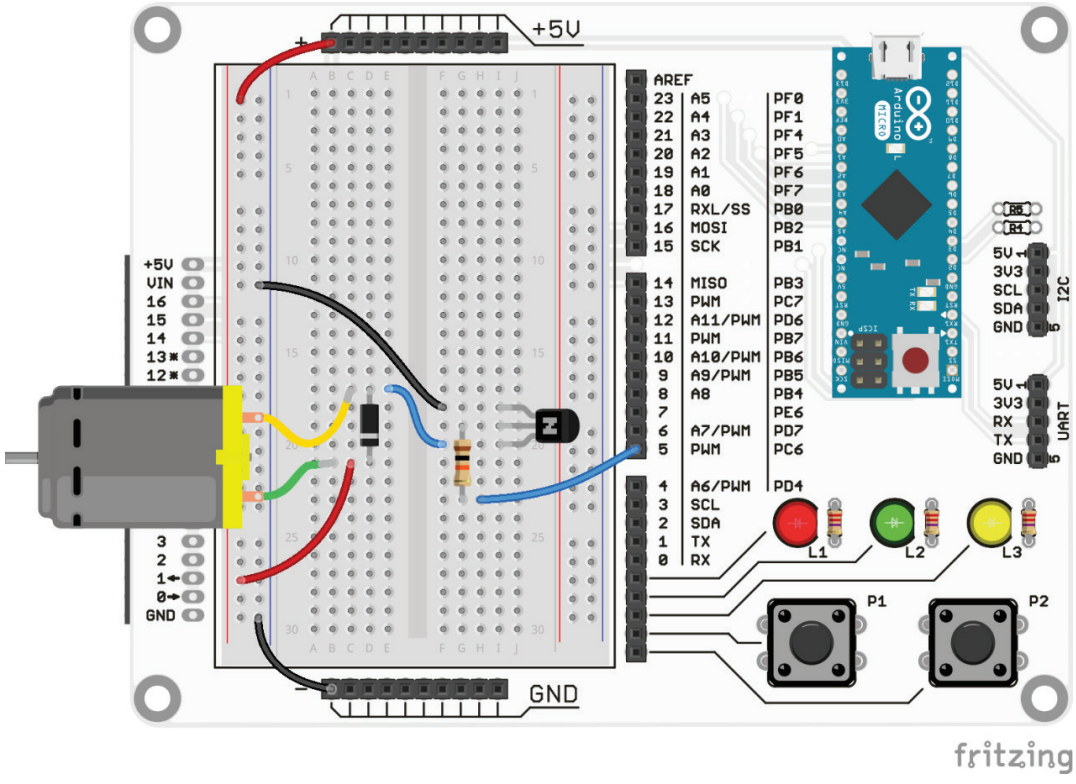
1. Saslēgt shēmu, kas redzama attēlā "Motora vadīšana ar programmu".
2. Pieslēgt shēmu sprieguma avotam. Ielādēt doto programmu.
3. Novērot, kad motors negriežas.
4. Saslēgt shēmu, kas redzama attēlā "Motora vadīšana ar programmu un tranzistoru".
5. Pieslēgt shēmu sprieguma avotam.
6. Novērot, kad motori griežas.

```
void setup ()
{
   pinMode(5,OUTPUT);
      //Uzstādām 5. MK kāju kā izeju
   #define motON digitalWrite(5,HIGH)
      //Definējam, kā ieslēgt pirmo diodi
   #define motOFF digitalWrite(5,LOW)
      //Definējam, kā izslēgt pirmo diodi
}
void loop ()
{
   motON;
}
```

Motora vadīšana ar programmu



Motora vadīšana ar programmu un tranzistoru

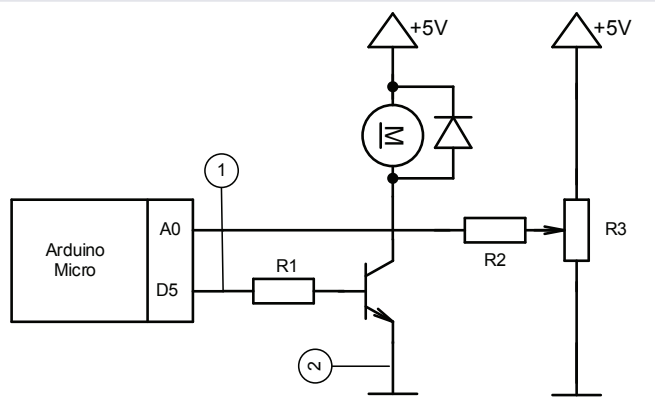


6.2. DARBA LAPA. Motoru ātruma regulēšana ar programmu

Mērķis

Apgūt PWM lietošanu motoru ātruma regulēšanā.

Izveidojamā shēma



Iepriekšējā uzdevuma shēma tiek papildināta ar potenciometru, kurš noteiks motora griešanās ātrumu.

Darba izpildes soļi

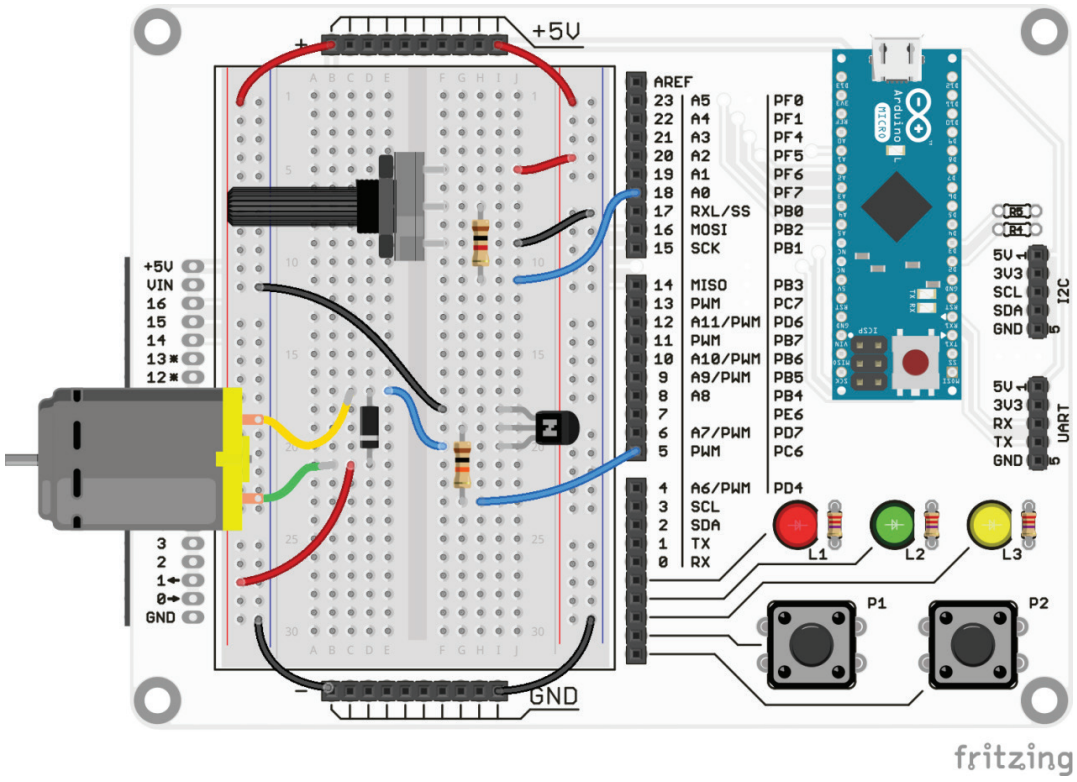
1. Saslēgt shēmu, kas redzama attēlā.
2. Uzrakstīt doto programmu.
3. Pieslēgt shēmu sprieguma avotam.
4. Ielādēt doto programmu.
5. Pagriezt potenciometra sviru.
6. Novērot, vai, pagriežot sviru, mainās motora ātrums.
7. Atvērt seriālā porta monitoru **Rīki → Seriālā porta monitors.**
8. Novērot, kā atkarībā no sviras pagriešanas leņķa mainās motora ātrums.
9. Pagriezt sviru tā, lai seriālā porta monitors rādītu aptuveni 200.
10. Nomērīt spriegumu starp punktiem 1 un 2.
11. Pagriezt sviru tā, lai seriālā porta monitors rādītu aptuveni 50.
12. Nomērīt spriegumu starp punktiem 1 un 2.
13. Noteikt, cik liels spriegums atbilst vienai seriālā porta rādītajai vērtībai.

```
int motPin = 5;   //Motora vadības izvads
int potPin = A0;
   //Potenciometru nolases ievads
void setup ()
{
   pinMode(motPin,OUTPUT);
      //Uzstādām motora vadības kāju ka izeju
      /*Analogās ieejas nolasa vērtības
      no 0 līdz 1023, bet motora vadība
      notiek vērtībās no 0 līdz 255, tāpēc
      sensoru nolasītā vērtība tiek izdalīta
      ar 4 */
   #define potReading analogRead(potPin)/4
      //Potenciometra nolasīšanas komandas
      //definīcija
   #define motDrive analogWrite(motPin,
                                potReading);
      //Motora vadīšanas definīcija
   Serial.begin(9600);
      //Ieslēdzam komunikāciju ar datoru
}
void loop ()
{
   motDrive;
      //Iedarbinām motoru atbilstoši
      //potenciometra rādījumam
   Serial.println(potReading);
      //Informācija par motora ātrumu
}
```

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors R1 (10 kΩ)	1 gab.
Rezistors R2 (1 kΩ)	1 gab.
Potenciometrs R3 (50 kΩ)	1 gab.
Līdzstrāvas motors (3–6 V)	1 gab.
Diode PH4148	1 gab.
NPN tranzistors BC517	1 gab.
Vads	Vairāki

Motora ātruma regulēšanas shēma



6.3. DARBA LAPA. Servomotora vadība

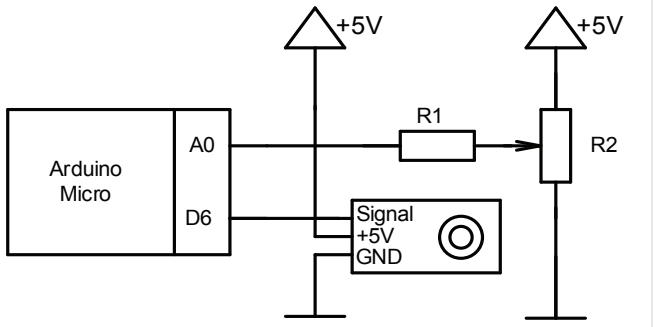
Mērķis

Apgūt servomotoru lietošanu un vadību ar programmu

Nepieciešamie materiāli

Materiāls/detaļa	Skaits
Rezistors R1 (1 kΩ)	1 gab.
Potenciometrs R2 (50 kΩ)	1 gab.
Līdzstrāvas servomotors (5 V)	1 gab.
Vads	Vairāki

Izveidojamā shēma



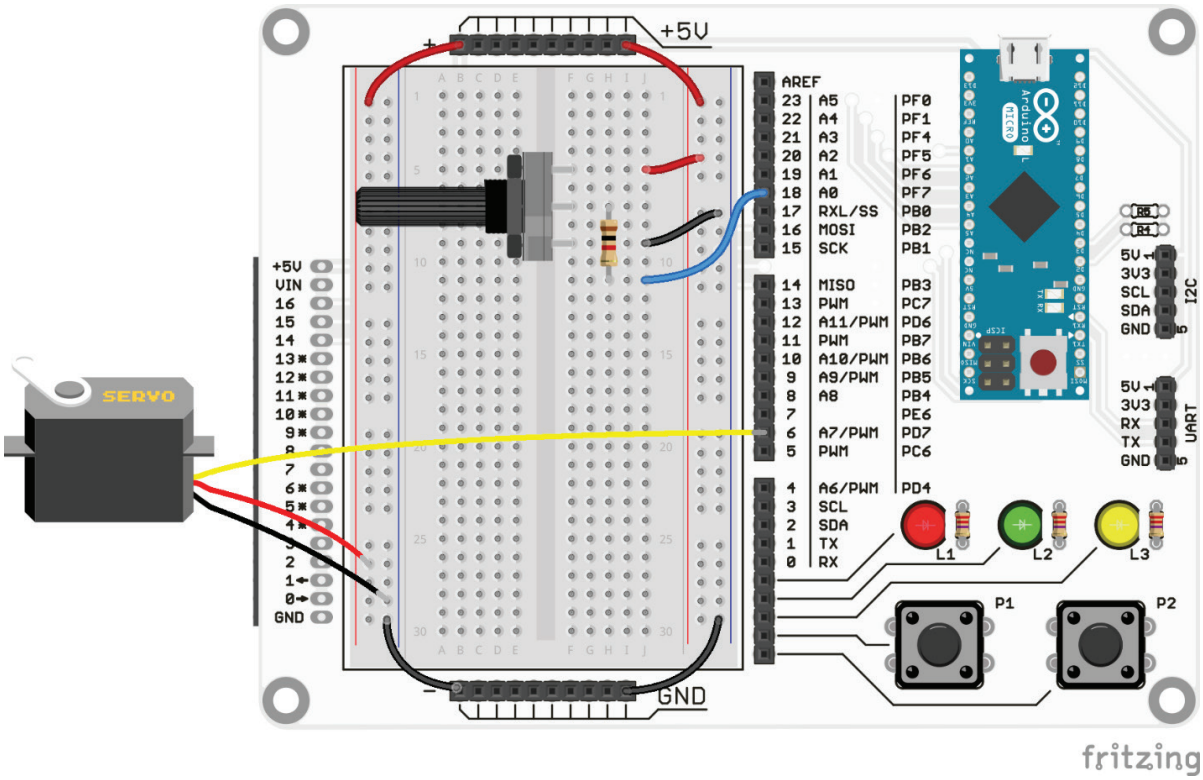
Servomotora pieslēgšanas shēma. Ir svarīgi pareizi pievienot motora izvadus. Spriegums nedrīkst pārsniegt 5 V.

Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā. Uzrakstīt šādu programmu.
2. Pieslēgt shēmu sprieguma avotam.
3. Ielādēt doto programmu.
4. Pagriezt potenciometra sviru.
5. Novērot, vai, pagriežot sviru, mainās servomotora pagriešanās leņķis.
6. Atvērt seriālā porta monitoru **Rīki → Seriālā porta monitors.**
7. Novērot, kā mainās motora leņķis atkarībā no sviras pagriešanas.

```
#include <Servo.h>
   //Pievieno bibliotēku servomotora vadībai
int servoPin = 6;
   //Servomotora vadības izvads
int potPin = A0;
   //Potenciometru nolases ievads
int servoAngle;
   //Servomotora pagriešanās leņķis
Servo servoMotor;
   //Izveido jaunu servomotora objektu
void setup ()
{
   servoMotor.attach(servoPin);
      //Pievieno servo objektam servo pinu
   #define potReading analogRead(potPin)
      //Potenciometra nolasīšanas komandas
      //definīcija
   Serial.begin(9600);
      //Ieslēdzam komunikāciju ar datoru
}
void loop ()
{
   servoAngle = map(potReading,0,1023,0,179);
      //Pārveido nolasītās potenciometra
      //vērtības grādos
   servoMotor.write(servoAngle);
      //Uzstāda uz servo pagriešanās leņķi
   Serial.println(servoAngle);
      //Informācija par servomotora leņķi
   delay(15);
      //Aizkave, lai servo nokļūtu savā
      //sākuma stāvoklī
}
```

Servomotora pieslēgšanas un vadības shēma



6.4. DARBA LAPA. *H* tilta lietošana

Mērķis

Apgūt *H* tilta mikroshēmas lietošanu.

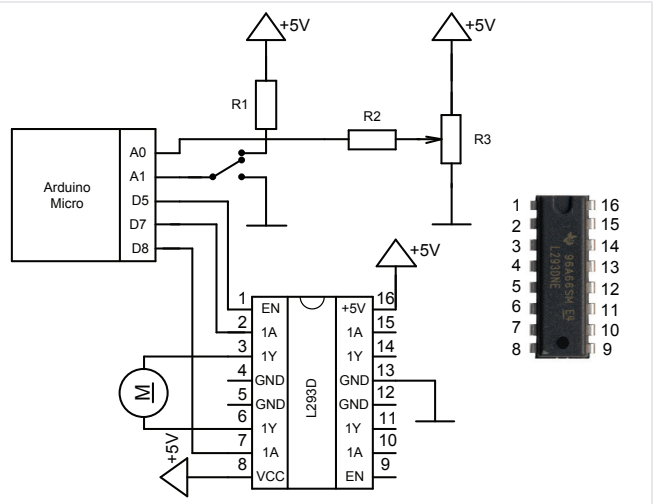
Darba izpildes soļi

1. Saslēgt shēmu, kas redzama attēlā.
2. Uzrakstīt doto programmu.
3. Ielādēt doto programmu.
4. Pagriezt potenciometra sviru, līdz motors sāk griezties.
5. Pārslēgt motora virziena slēdzi.
6. Novērot motora griešanās virziena izmaiņu.
7. Pieslēgt shēmu sprieguma avotam.

Nepieciešamie materiāli

Materiāls/detaļa	Nomināls
H tilta mikroshēma L293D	1 gab.
Rezistors R1 (10 kΩ)	1 gab.
Rezistors R2 (1 kΩ)	1 gab.
Potenciometrs R3 (50 kΩ)	1 gab.
Slēdzis ON-ON	1 gab.
Līdzstrāvas motors (3–6 V)	1 gab.
Vads	Vairāki

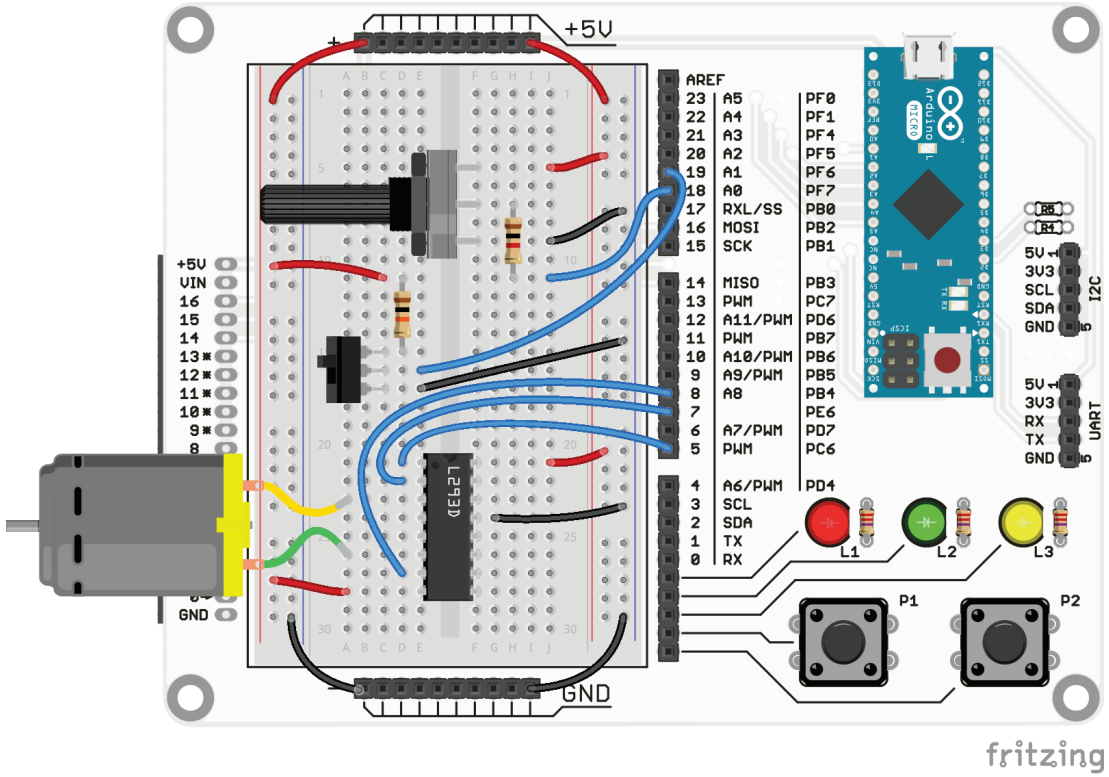
Izveidojamā shēma



H tilta mikroshēmas pieslēgšana viena motora vadībai

```
int dirPin1 =7;      //Virziena pins 1
int dirPin2 =8;      //Virziena pins 2
int speedPin = 5;    //Motora ātruma izvads
int potPin = A0;
   //Potenciometru nolases ievads
int switchPin = A1;
   //Virziena slēdža nolases pins

void setup ()
{
   pinMode(dirPin1,OUTPUT);
      //Uzstāda virziena 1. pinu kā izeju
   pinMode(dirPin2,OUTPUT);
      //Uzstāda virziena 2. pinu kā izeju
   pinMode(speedPin,OUTPUT);
      //Uzstāda ātruma pinu kā izeju
      /*Analogās ieejas nolasa vērtības
      no 0 līdz 1023, bet motora vadība
      notiek no 0 līdz 255,
      tāpēc sensoru nolasītā vērtība tiek
      izdalīta ar 4 */
   #define potReading analogRead(potPin)/4
      //Potenciometra nolasīšanas komandas
      //definīcija
   #define switchReading digitalRead(switchPin)
      //Slēdža nolasīšanas komandas definīcija
   digitalWrite(switchPin,HIGH);
      //Uzstāda slēdža pinam iekšējo pull-up
      //rezistoru
}
void MotorDrive (int motSpeed,int motDirec-
                                    tion)
   //Motora vadības funkcija
{
   analogWrite(speedPin,motSpeed);
      //Motora ātruma uzstādīšana
   if (motDirection)
      //Motora virziena uzstādīšana
{
   digitalWrite(dirPin1,HIGH);
   digitalWrite(dirPin2,LOW);
}
   else
   {
      digitalWrite(dirPin1,LOW);
      digitalWrite(dirPin2,HIGH);
   }
}
void loop ()
{
   MotorDrive(potReading,switchReading);
      //Izsauc motora vadības funkciju
}
```

H tilta mikroshēmas lietošana



7. MINISUMO ROBOTS

7.1. Mērķis

Nodaļas mērķis ir sniegt ieskatu par konstruktorā ietvertā robota darbību un tā programmēšanas īpatnībām, virzoties pretī pilnvērtīga minisumo robota izstrādei un konkurētspējīgas programmatūras izveidei. Tādēļ te aplūkota robota uzbūve un to pamatelementu nozīme, tiks izstrādāta piemēra jeb testa programma robota mezglu darbības pārbaudei, kas kalpos par bāzi turpmākām sarežģītākām programmām.

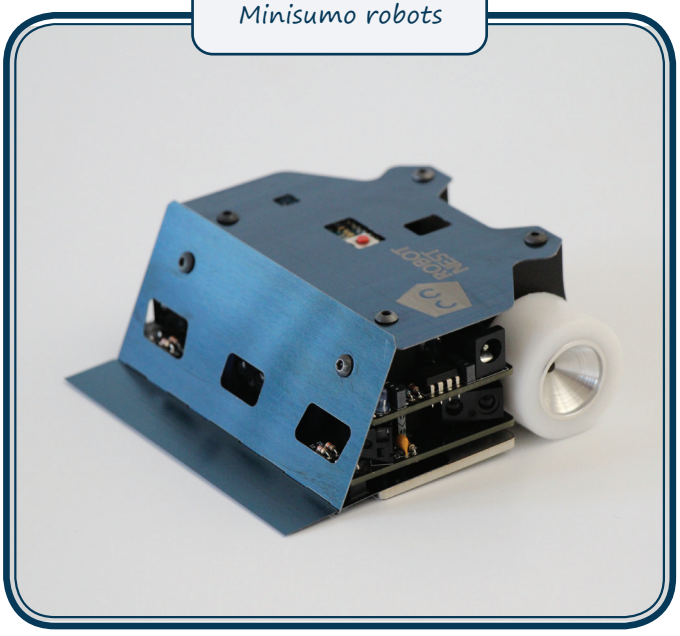
7.2. Teorija

Lielāko daļu robotu veido trīs galvenās daļas, kurām darbojoties saskaņoti var panākt robota augstu veiktspēju un iecerēto mērķu sasniegšanu. Šīs daļas ir:

- **Mehānika** — tas ir viss fiziskais robotā, ko varam paņemt rokā, — korpuss, iespiedplates, motori, riteņi, akumulators u. c. Veidojot mehāniku, ir jāievēro minisumo sacensību noteikumi, lai drīkstētu piedalīties starptautiska līmeņa sacensībās Latvijā vai ārpus tās. Atbilstoši minisumo sacensību noteikumiem minisumo robots nevar būt lielāks par 10 × 10 cm un svērt ne vairāk kā 500 g. Tā komponentes nedrīkst bojāt laukuma virsmu, atdalīties, ietekmēt pretinieku darbību. Ja tiek ievēroti sacensību noteikumi, konstrukciju var veidot tādu, kādu vēlas. Tomēr ir jāpatur prātā, ka eksistē arī cita pielietojuma roboti, kuru mehānisko komponenšu izgatavošana un izstrāde prasa daudz laika un uzmanības, jo tikai visas šeit minētās robota daļas sadarbībā nodrošina tā veiksmīgu darbību.
- **Elektronika** ir līdzeklis, kas robotā visu sasaista vienotā sistēmā. Tā robota programmai piegādā datus no sensoriem un pēc programmas komandām kustina robota mehāniku. Tā tieši ietekmē, kā robots būs jāprogrammē un ko robots spēs izdarīt. Šī konstruktora robota elektronika ir parādīta shēmā attēlā "Robota principiālā shēma — sensori un motoru vadība, procesors un tā palīgshēmas".
- **Programmatūra** ir līdzeklis, kas ļauj realizēt paša izdomātu robota darbību. Tādēļ bieži sacensībās uzvar gudrākais, nevis stiprākais robots. Tā apstrādā sensoru informāciju un dod pavēles robota izvadierīcēm, kā arī motoriem un mirdzdiodēm. Robots darīs tikai to, kas norādīts tā programmā. Tāpēc, ja robots nepilda savu uzdevumu, visbiežāk ir jāmeklē kļūda programmā. Visa robota programma glabājās mikrokontrollera atmiņā, kas atrodas uz Arduino plates.

Ja visas daļas darbojas saskaņoti un pareizi, robots kustas un veic savu uzdevumu atbilstoši plānotajam. Pareizas un efektīvas programmas izstrāde nav vienkārša, jo ir jāņem vērā gan mehānikas, gan elektronikas īpatnības. Šajā nodaļā ir aplūkoti šādas programmas izveides pamati, kas jums ļaus izveidot savu programmu. Tādu, ar ko jūs varēsiet uzvarēt.

Minisumo robots



7.1. DARBA LAPA. Robota mirdzdiodi

Mērķis

Apgūt minisumo robota "SumoBoy" mirdzdiožu vadību.

Robotā izmantotās ierīces

Numurs diagrammā	Nosaukums programmā	Arduino Pins
14	ledA	13
3	led1	5
8	led2	11
16	led3	2

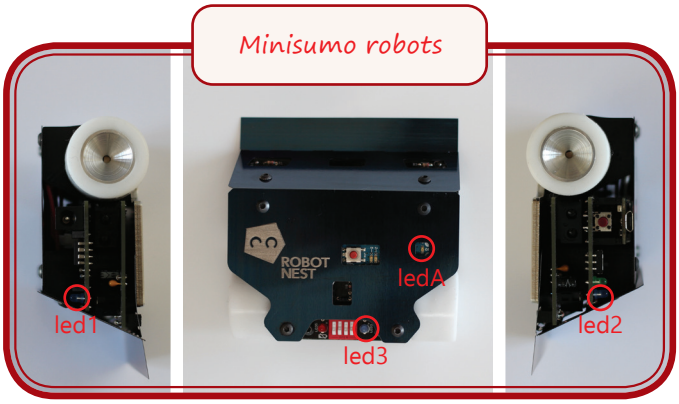
Uzdevumi

1. Izveidot diodes mirkšķināšanas programmu diodei ledA.

```
void setup() {   //Uzstādīšana
   pinMode(13,OUTPUT);
      //Definē 13. kāju kā izeju
#define ledAON digitalWrite(13,HIGH)
   //Definīcija ledA ieslēgšanai
#define ledAOFF digitalWrite(13,LOW)
   //Definīcija ledA izslēgšanai
}
void loop() {   //Mūžīgais cikls
   ledAON;   //Ieslēdz ledA
   delay(1000);   //Pagaida 1000 milisekundes
   ledAOFF;   //Izslēdz ledA
   delay(1000);   //Pagaida 1000 ms
}
```

2. Pieslēgt robotu datora USB pieslēgvietai.
3. Augšupielādēt programmu robotā.
4. Novērot, kā mirkšķinās ledA diode.
5. Pamainiet gaidīšanas laikus, lai diode mirkšķinās ātrāk.
6. Saglabājiet programmu, turpmāk papildināsim un mainīsim esošo kodu.
7. Izveidojiet programmu, kur visas robota diodes pēc kārtas ieslēdzas un izslēdzas, papildinot iepriekšējo programmas fragmentu. Papildinājumi izcelti. Augšupielādēt programmu robotā.
8. Novērot, kā diodes pēc kārtas ieslēdzas un izslēdzas.

```
void setup() {   //Uzstādīšana
//LED definīcijas
  pinMode(13,OUTPUT);   //Arduino LED
#define ledAON  digitalWrite(13,HIGH)
#define ledAOFF digitalWrite(13,LOW)
  pinMode(5,OUTPUT);   //Left LED
#define led1ON  digitalWrite(5,HIGH);
#define led1OFF digitalWrite(5,LOW);
  pinMode(11,OUTPUT);   //Right LED
#define led2ON  digitalWrite(11,HIGH);
#define led2OFF digitalWrite(11,LOW);
  pinMode(2,OUTPUT);   //Back LED
#define led3ON  digitalWrite(2,HIGH);
#define led3OFF digitalWrite(2,LOW);
}
void loop() {   //Mūžīgais cikls
  ledAON;   //Ieslēdz ledA
  delay(1000);   //Pagaida 1000 ms
  ledAOFF;   //Izslēdz ledA
  led1ON;   //Ieslēdz led1
  delay(1000);   //Pagaida 1000 ms
  led1OFF;   //Izslēdz led1
  led2ON;
  delay(1000);
  led2OFF;
  led3ON;
  delay(1000);
  led3OFF;
}
```



7.2. DARBA LAPA. Robota pogas

Mērķis

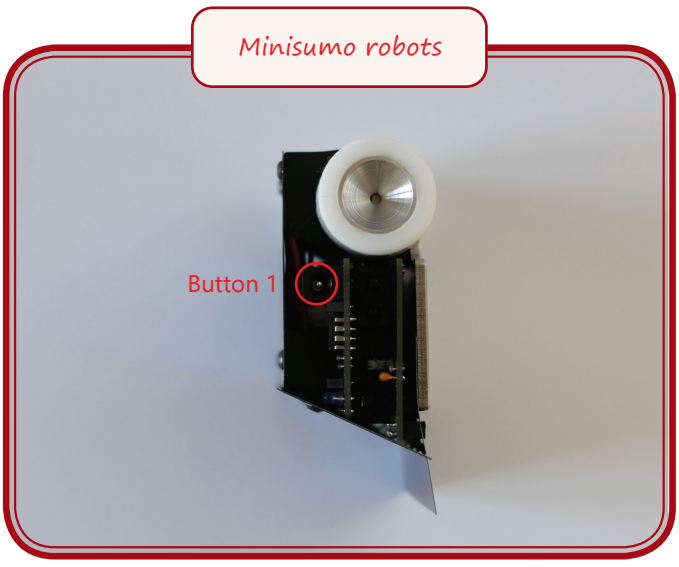
Apgūt minisumo robota spiedpogu darbību un seriālā porta monitora lietošanu.

Robotā izmantotās ierīces

Numurs diagrammā	Nosaukums programmā	Arduino MK kāja
12	Button1	12

Uzdevumi

1. Papildināt iepriekš izveidoto programmu, kur BUTTON1 ieslēdz LED mirkšķināšanos un izslēdz, kā arī izvada uz datora informāciju par pogas stāvokli.
2. Pieslēgt robotu datora USB pieslēgvietai.
3. Augšupielādēt programmu robotā.
4. Nospiest pogas un pārbaudīt, vai ar tām var izslēgt uz ieslēgt diožu mirkšķināšanos.
5. Ieslēgt virknes pieslēguma monitoru un pārbaudīt, vai parādās paziņojumi par pogu nospiešanu.
6. Saglabājiet programmu, turpmāk papildināsim un mainīsim esošo kodu




```
void setup() {   //Uzstādīšana
   //LED definīcijas
  pinMode(13,OUTPUT);   //Arduino LED
#define ledAON   digitalWrite(13,HIGH)
#define ledAOFF  digitalWrite(13,LOW)
  pinMode(5,OUTPUT);   //Left LED
#define led1ON   digitalWrite(5,HIGH);
#define led1OFF  digitalWrite(5,LOW);
  pinMode(11,OUTPUT);   //Right LED
#define led2ON   digitalWrite(11,HIGH);
#define led2OFF  digitalWrite(11,LOW);
  pinMode(2,OUTPUT);   //Back LED
#define led3ON   digitalWrite(2,HIGH);
#define led3OFF  digitalWrite(2,LOW);
   //Pogu definīcijas
  pinMode(12,INPUT);
  digitalWrite(12,HIGH);
#define button1 !digitalRead(12)
   //Seriālā porta monitora ieslēgšana
  Serial.begin(9600);
}
void loop() {   //Mūžīgais cikls
  led1OFF;
  led2OFF;
  led3OFF;
   //Izslēdz visas diodes programmas sākumā
  if (button1)   //Ieslēgs diožu mirkšķināšanu,
                 //ja poga 1 nospiesta
  {
    while (button1){
   //Cikls, kamēr poga 1 ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
   //Izvada, kad nospiesta poga 1
    }
    while(!button1)
   //Mirkšķina diodes, kamēr poga 1 netiek
   //nospiesta atkārtoti
    {
      led1ON;
      led2ON;
      led3ON;   //Ieslēdz trīs diodes
      delay(1000);   //Pagaida 1000 ms
      led1OFF;
      led2OFF;
      led3OFF;   //Izslēdz trīs diodes
      delay(1000);   //Pagaida 1000 ms
    }
    while (button1){
   //Cikls, kamēr poga 1 ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
   //Izvada, kad nospiesta poga 1
    }
  }
}
```

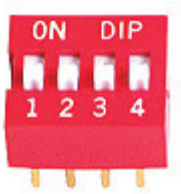
7.3. DARBA LAPA. Robota DIP slēdži

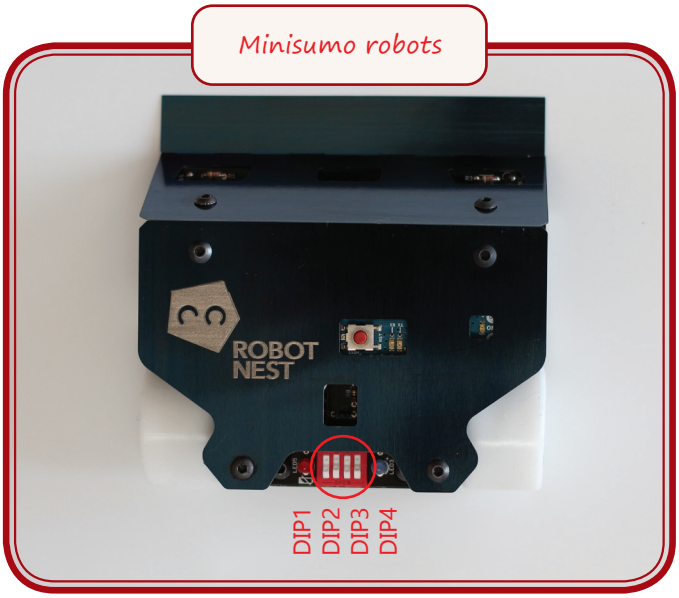
Mērķis

Apgūt robota DIP slēdžu lietošanu, kas ļauj iedarbināt dažādas robota rīcības stratēģijas.

Robotā izmantotās ierīces

Numurs diagrammā	Nosaukums programmā	Arduino Pins
23	DIP1	15
23	DIP2	14
23	DIP3	16
23	DIP4	1





Uzdevumi

1. Papildināt pogu programmu, lai atkarībā no DIP slēdžu kombinācijām mainītos mirkšķināšanās ātrums
2. Pieslēgt robotu datora USB pieslēgvietai.
3. Augšupielādēt programmu robotā.
4. Nospiest pogas un pārbaudīt, vai ar tām var izslēgt uz ieslēgt diožu mirkšķināšanos.
5. Izmēģināt dažādas DIP slēdža kombinācijas un pārbaudīt mirkšķināšanās ātruma izmaiņas.
6. Saglabājiet programmu, turpmāk papildināsim un mainīsim esošo kodu.

DIP1	DIP2	DIP3	DIP4	Ātruma reizinātājs
OFF	OFF	OFF	OFF	× 0
OFF	OFF	OFF	ON	× 1
OFF	OFF	ON	OFF	× 2
OFF	OFF	ON	ON	× 3
OFF	ON	OFF	OFF	× 4
OFF	ON	OFF	ON	× 5
OFF	ON	ON	OFF	× 6
OFF	ON	ON	ON	× 7
ON	OFF	OFF	OFF	× 8
ON	OFF	OFF	ON	× 9
ON	OFF	ON	OFF	× 10
ON	OFF	ON	ON	× 11
ON	ON	OFF	OFF	× 12
ON	ON	OFF	ON	× 13
ON	ON	ON	OFF	× 14
ON	ON	ON	ON	× 15

```
void setup() {   //Uzstādīšana
   //LED definīcijas
  pinMode(13,OUTPUT);   //Arduino LED
#define ledAON   digitalWrite(13,HIGH)
#define ledAOFF  digitalWrite(13,LOW)
  pinMode(5,OUTPUT);   //Left LED
#define led1ON   digitalWrite(5,HIGH);
#define led1OFF  digitalWrite(5,LOW);
  pinMode(11,OUTPUT);   //Right LED
#define led2ON   digitalWrite(11,HIGH);
#define led2OFF  digitalWrite(11,LOW);
  pinMode(2,OUTPUT);   //Back LED
#define led3ON   digitalWrite(2,HIGH);
#define led3OFF  digitalWrite(2,LOW);
   //Pogu definīcijas
  pinMode(12,INPUT);
  digitalWrite(12,HIGH);
#define button1 !digitalRead(12)
   //Seriālā porta monitora ieslēgšana
  Serial.begin(9600);
     //DIP slēdžu definīcija
  pinMode(15,INPUT);
  digitalWrite(15,HIGH);
#define DIP1 digitalRead(15)
  pinMode(14,INPUT);
  digitalWrite(14,HIGH);
#define DIP2 digitalRead(14)
  pinMode(16,INPUT);
  digitalWrite(16,HIGH);
#define DIP3 digitalRead(16)
  pinMode(17,INPUT);
  digitalWrite(17,HIGH);
#define DIP4 digitalRead(17)
}
int switchCounter =0;
   //Izveido mainīgo nospiesto slēdžu
   //saskaitīšanai
int blinkingSpeed =0;
   //Izveido mainīgo mirkšķināšanās ātrumam
void loop() {   //Mūžīgais cikls
digitalWrite(17,HIGH);
   //Uzstāda augstu līmeni, lai var nolasit
   //4. slēdzi
  led1OFF;
  led2OFF;
  led3OFF;
     //Izslēdz visas diodes programmas sākumā
  if (button1)
     //Ieslēgs diožu mirkšķināšanu,
     //ja poga 1 nospiesta
  {
    while (button1){
      //Cikls, kamēr poga ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
    while(!button1)
      //Mirkšķina diodes, kamēr poga 1 netiek
      //nospiesta
    {
      switchCounter = 0;
        //Nonullē slēdžu skaitītāju
      if(DIP1){
        switchCounter =switchCounter+8;
      }   //Ja slēdzis 1 ieslēgts, pieskaita 8
      if(DIP2){
        switchCounter =switchCounter+4;
      }
      if(DIP3){
        switchCounter =switchCounter+2;
      }
      if(DIP4){
        switchCounter =switchCounter+1;
      }
      blinkingSpeed =(switchCounter*100)+100;
         //Aprēķina mirkšķināšanās ātrumu
      led1ON;
      led2ON;
      led3ON;   //Ieslēdz trīs diodes
      delay(blinkingSpeed);
        //Pagaida mirkšķināšanās ātrumu
      led1OFF;
      led2OFF;
      led3OFF;   //Izslēdz trīs diodes
      delay(blinkingSpeed);
        //Pagaida mirkšķināšanās ātrumu
    }
    while (button1){
      //Cikls, kamēr poga 1 ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
  }
}
```

7.4. DARBA LAPA. Robota līnijas sensori

Mērķis

Apgūt minisumo līnijas sensoru lietošanu.

Robotā izmantotās ierīces

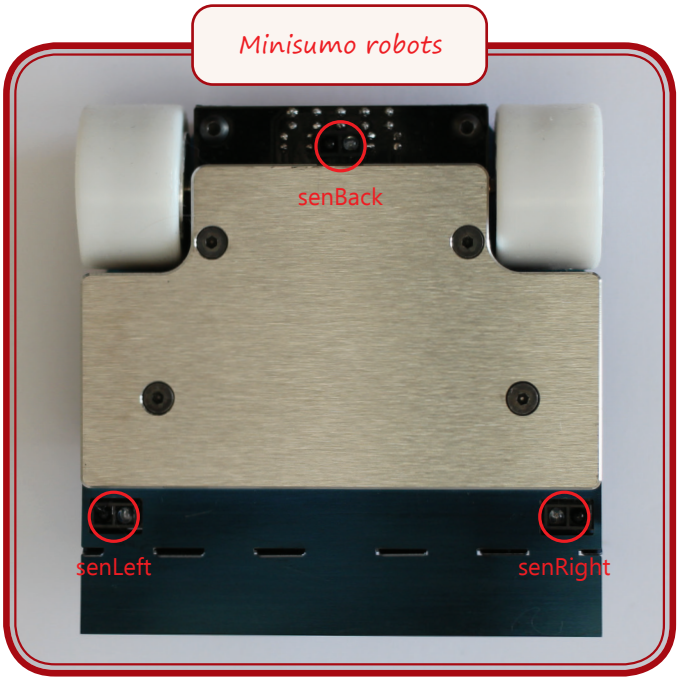
Numurs diagrammā	Nosaukums programmā	Arduino MK kājas
19	senRight	A1
20	senLeft	A2
21	senBack	A4

Uzdevumi

1. Papildināt programmu, lai robota līnijas sensoram, nonākot uz baltas virsmas, iedegtos kāda no mirdzdiodēm.

Diode	Līnijas sensors
led1	senLeft
led2	senRight
led3	senBack

2. Pieslēgt robotu USB pieslēgvietai.
3. Augšupielādēt programmu robotā.
4. Uzlikt robotu uz gaišas virsmas un pārbaudīt, vai visas diodes deg.
5. Ja kāda diode nedeg, tad jāmaina sensora nostrādes robežvērtības.
6. Pēc kārtas uzlikt katru sensoru uz tumšas virsmas un pārbaudīt, vai diode izdziest.
7. Saglabājiet programmu, turpmāk papildināsim un mainīsim esošo kodu.



```
void setup() {   //Uzstādīšana
   //LED definīcijas
  pinMode(13,OUTPUT);   //Arduino LED
#define ledAON   digitalWrite(13,HIGH)
#define ledAOFF  digitalWrite(13,LOW)
  pinMode(5,OUTPUT);   //Left LED
#define led1ON   digitalWrite(5,HIGH);
#define led1OFF  digitalWrite(5,LOW);
  pinMode(11,OUTPUT);   //Right LED
#define led2ON   digitalWrite(11,HIGH);
#define led2OFF  digitalWrite(11,LOW);
  pinMode(2,OUTPUT);   //Back LED
#define led3ON   digitalWrite(2,HIGH);
#define led3OFF  digitalWrite(2,LOW);
   //Pogu definīcijas
  pinMode(12,INPUT);
  digitalWrite(12,HIGH);
#define button1 !digitalRead(12)
   //Seriālā porta monitora ieslēgšana
  Serial.begin(9600);
     //DIP slēdžu definīcija
  pinMode(15,INPUT);
  digitalWrite(15,HIGH);
#define DIP1 digitalRead(15)
  pinMode(14,INPUT);
  digitalWrite(14,HIGH);
#define DIP2 digitalRead(14)
  pinMode(16,INPUT);
  digitalWrite(16,HIGH);
#define DIP3 digitalRead(16)
  pinMode(17,INPUT);
  digitalWrite(17,HIGH);
#define DIP4 digitalRead(17)
   //Līnijas sensoru definīcija
  pinMode(A2,INPUT);
#define senLeft  analogRead(A2)<50
  pinMode(A1,INPUT);
#define senRight analogRead(A1)<50
  pinMode(A4,INPUT);
#define senBack  analogRead(A4)<50
}
void loop() {   //Mūžīgais cikls
  led1OFF;
  led2OFF;
  led3OFF;
     //Izslēdz visas diodes programmas sākumā
  if (button1)
     //Ieslēgs diožu mirkšķināšanu,
     //ja poga 1 nospiesta
  {
   while (button1){
      //Cikls, kamēr poga 1 ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
    while(!button1)
      //Mirkšķina diodes, kamēr poga 1
      //netiek nospiesta
    {
      if (senLeft){led1ON;}
        //Ja kreisais sensors redz baltu,
        //tad diode ieslēdzas
      else {led1OFF;}   //Citādi diode izslē-
                          dzas
      if (senRight){led2ON;}
      else {led2OFF;}
      if (senBack){led3ON;}
      else {led3OFF;}
    }
    while (button1){
      //Cikls, kamēr poga ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
  }
```

7.5. DARBA LAPA. Robota attāluma sensori

Mērķis

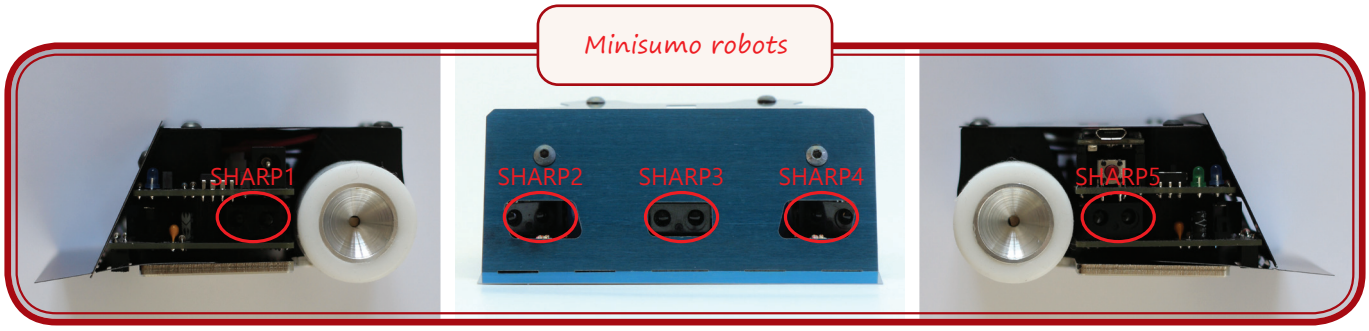
Apgūt minisumo SHARP attāluma sensoru lietošanu.

Robotā izmantotās ierīces

Numurs diagrammā	Nosaukums programmā	Arduino Pins
2	SHARP1	A5
5	SHARP2	A3
6	SHARP3	7
7	SHARP4	A8
11	SHARP5	A7

Uzdevumi

1. Papildināt programmu, lai, iedarbojoties SHARP sensoram, monitorā parādītos atbilstošs paziņojums.
2. Pieslēgt robotu datora USB pieslēgvietai.
3. Novērot, vai pat bez programmas darbības, diodes pie SHARP sensoriem iedegas, ja SHARP sensoram pieliek priekšā roku.
4. Augšupielādēt programmu robotā.
5. Ieslēgt virknes pieslēguma monitoru.
6. Pārbaudīt, kad, pieliekot roku pie attiecīgā sensora, parādās paziņojums.
7. Saglabājiet programmu, turpmāk papildināsim un mainīsim esošo kodu.



```
void setup() {   //Uzstādīšana
   //LED definīcijas
  pinMode(13,OUTPUT);   //Arduino LED
#define ledAON   digitalWrite(13,HIGH)
#define ledAOFF  digitalWrite(13,LOW)
  pinMode(5,OUTPUT);   //Left LED
#define led1ON   digitalWrite(5,HIGH);
#define led1OFF  digitalWrite(5,LOW);
  pinMode(11,OUTPUT);   //Right LED
#define led2ON   digitalWrite(11,HIGH);
#define led2OFF  digitalWrite(11,LOW);
  pinMode(2,OUTPUT);   //Back LED
#define led3ON   digitalWrite(2,HIGH);
#define led3OFF  digitalWrite(2,LOW);
   //Pogu definīcijas
  pinMode(12,INPUT);
  digitalWrite(12,HIGH);
#define button1 !digitalRead(12)
   //Seriālā porta monitora ieslēgšana
  Serial.begin(9600);
     //DIP slēdžu definīcija
  pinMode(15,INPUT);
  digitalWrite(15,HIGH);
#define DIP1 digitalRead(15)
  pinMode(14,INPUT);
  digitalWrite(14,HIGH);
#define DIP2 digitalRead(14)
  pinMode(16,INPUT);
  digitalWrite(16,HIGH);
#define DIP3 digitalRead(16)
  pinMode(17,INPUT);
  digitalWrite(17,HIGH);
#define DIP4 digitalRead(17)
   //Līnijas sensoru definīcija
  pinMode(A2,INPUT);
#define senLeft  analogRead(A2)<50
  pinMode(A1,INPUT);
#define senRight analogRead(A1)<50
  pinMode(A4,INPUT);
#define senBack  analogRead(A4)<50
   //SHARP sensoru definīcija
  pinMode(A5,INPUT);
#define SHARP1 !digitalRead(A5)
  pinMode(A3,INPUT);   //SHARP2
#define SHARP2 !digitalRead(A3)
  pinMode(7,INPUT);   //SHARP3
#define SHARP3 !digitalRead(7)
  pinMode(A8,INPUT);   //SHARP4
#define SHARP4 !digitalRead(A8)
  pinMode(A7,INPUT);   //SHARP5
#define SHARP5 !digitalRead(A7)
}
void loop() {   //Mūžīgais cikls
  led1OFF;
  led2OFF;
  led3OFF;
     //Izslēdz visas diodes programmas sākumā
  if (button1)
     //Ieslēgs diožu mirkšķināšanu,
     //ja poga 1 nospiesta
  {
   while (button1){
     //Cikls, kamēr poga ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
    while(!button1)
      //Mirkšķina diodes, kamēr poga 1
      //netiek nospiesta
    {
     if (SHARP1){Serial.println("Skerslis
                              kreisaja mala");}
       //Ja ir šķērslis, tad tiek izvadīts
       //paziņojums
     if (SHARP2){Serial.println("Skerslis
                           labaja puse vidu");}
     if (SHARP3){Serial.println("Skerslis
                                    vidu");}
     if (SHARP4){Serial.println("Skerslis
                        kreisaja puse vidu");}
     if (SHARP5){Serial.println("Skerslis
                               labaja mala");}
     delay(500);
       //Pussekundes aizkave, lai tekstu
       //varetu izlasīt
    }
    while (button1){
      //Cikls, kamēr poga ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
  }
}
```

7.6. DARBA LAPA. Robota motori

Mērķis

Apgūt minisumo motoru lietošanu.

Robotā izmantotās ierīces

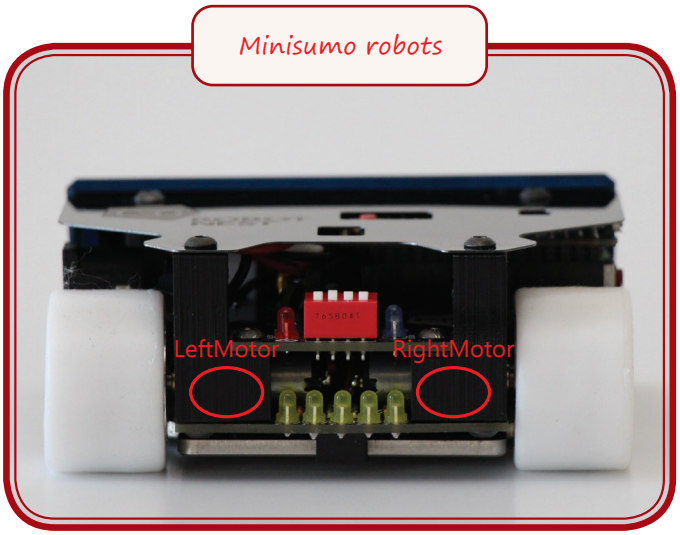
Numurs diagrammā	Nosaukums	Arduino Pins
17	Labā motora vadības pins	9
17	Labā motora vadības pins	10
22	Kreisā motora vadības pins	3
22	Kreisā motora vadības pins	4

Uzdevumi

1. Papildināt programmu, lai, ieslēdzot vienu no slēdžiem, robots brauktu uz dažādos virzienos.

Motora darbība	Kreisā motora griešanās virziens	Labā motora griešanās virziens
Forward	↑	↑
Backward	↓	↓
Left	↓	↑
Right	↑	↓

2. Pieslēgt robotu datora USB pieslēgvietai.
3. Augšupielādēt programmu robotā.
4. Ieslēgt tikai vienu no slēdžiem.
5. Nospiest pogu 1.
6. Novērot, vai motori griežas.
7. Pārbaudīt visus robota braukšanas virzienus.
8. Ja nospiesta vairāk nekā viena poga, tad robots izpildīs pēdējās pogas darbību. Pārbaudiet to.
9. Saglabājiet programmu, turpmāk papildinā-sim un mainīsim esošo kodu.



```
void setup() {   //Uzstādīšana
//LED definīcijas
  pinMode(13,OUTPUT);   //Arduino LED
#define ledAON   digitalWrite(13,HIGH)
#define ledAOFF  digitalWrite(13,LOW)
  pinMode(5,OUTPUT);   //Left LED
#define led1ON   digitalWrite(5,HIGH);
#define led1OFF  digitalWrite(5,LOW);
  pinMode(11,OUTPUT);   //Right LED
#define led2ON   digitalWrite(11,HIGH);
#define led2OFF  digitalWrite(11,LOW);
  pinMode(2,OUTPUT);   //Back LED
#define led3ON   digitalWrite(2,HIGH);
#define led3OFF  digitalWrite(2,LOW);
   //Pogu definīcijas
  pinMode(12,INPUT);
  digitalWrite(12,HIGH);
#define button1 !digitalRead(12)
   //Seriālā porta monitora ieslēgšana
  Serial.begin(9600);
     //DIP slēdžu definīcija
  pinMode(15,INPUT);
  digitalWrite(15,HIGH);
#define DIP1 digitalRead(15)
  pinMode(14,INPUT);
  digitalWrite(14,HIGH);
#define DIP2 digitalRead(14)
  pinMode(16,INPUT);
  digitalWrite(16,HIGH);
#define DIP3 digitalRead(16)
  pinMode(17,INPUT);
  digitalWrite(17,HIGH);
#define DIP4 digitalRead(17)
   //Līnijas sensoru definīcija
  pinMode(A2,INPUT);
#define senLeft  analogRead(A2)<50
  pinMode(A1,INPUT);
#define senRight analogRead(A1)<50
  pinMode(A4,INPUT);
#define senBack  analogRead(A4)<50
   //SHARP sensoru definīcija
  pinMode(A5,INPUT);
#define SHARP1 !digitalRead(A5)
  pinMode(A3,INPUT);//SHARP2
#define SHARP2 !digitalRead(A3)
  pinMode(7,INPUT);//SHARP3
#define SHARP3 !digitalRead(7)
  pinMode(A8,INPUT);//SHARP4
#define SHARP4 !digitalRead(A8)
  pinMode(A7,INPUT);//SHARP5
#define SHARP5 !digitalRead(A7)
}
void loop() {   //Mūžīgais cikls
  led1OFF;
  led2OFF;
  led3OFF;
     //Izslēdz visas diodes programmas sākumā
  if (button1)
     //Ieslēgs diožu mirkšķināšanu,
     //ja poga 1 nospiesta
  {
   while (button1){
     //Cikls, kamēr poga ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
    while(!button1)
      //Mirkšķina diodes, kamēr poga 1
      //netiek nospiesta
    {
     if (SHARP1){Serial.println("Skerslis
                                kreisaja mala");}
       //Ja ir šķērslis, tad tiek izvadīts
       //paziņojums
     if (SHARP2){Serial.println("Skerslis
                              labaja puse vidu");}
     if (SHARP3){Serial.println("Skerslis
                                      vidu");}
     if (SHARP4){Serial.println("Skerslis
                           kreisaja puse vidu");}
     if (SHARP5){Serial.println("Skerslis
                                labaja mala");}
     delay(500);
       //Pussekundes aizkave, lai tekstu
       //varetu izlasīt
    }
    while (button1){
      //Cikls, kamēr poga ir nospiesta
    Serial.println("Poga 1 ir nospiesta");
      //Izvada, kad nospiesta poga 1
    }
  }
}
```

8. MINISUMO SACENSĪBAS

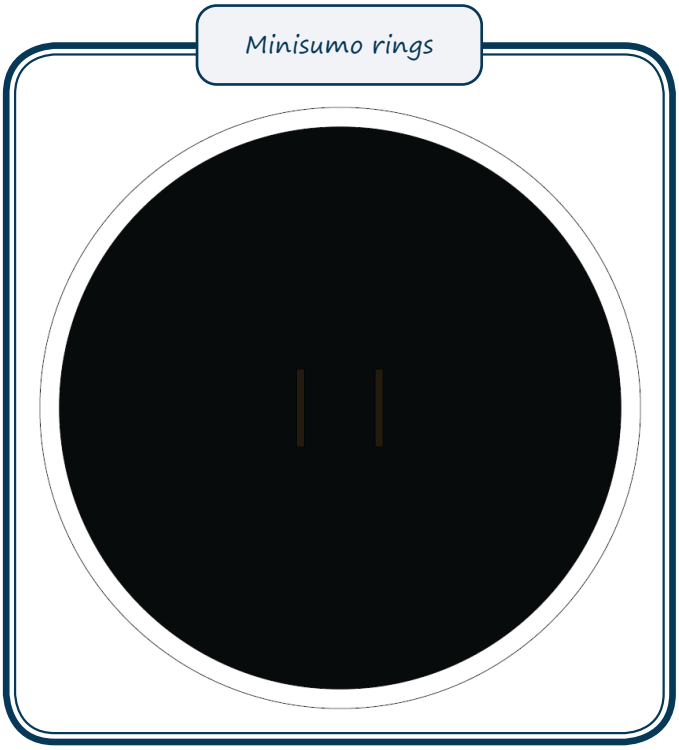
8.1. Mērķis

Šajā nodaļā īsi tiks apskatīti minisumo sacensību noteikumi. Balstoties uz tiem, tiks veidota vienkārša minisumo cīņas programma, ar kuru jūs varēsiet piedalīties starptautiska līmeņa minisumo sacensībās.

8.2. Minisumo sacensību noteikumi

Minisumo ir robotu sacensību disciplīna, kas radusies, balstoties uz Japānas sumo cīņām. Robotu sumo cīņās parasti piedalās divi roboti uz apaļa ringa. Roboti autonomi cīnās uz ringa, līdz kāds no tiem tiek izgrūsts ārpus ringa. Uzvarētājs ir tas robots, kurš palicis ringā.

Minisumo sacensību rings — melns aplis 77 cm diametrā ar 2,5 cm platu baltu līniju apkārt tam. Ringa centrā ir divas, tumši brūnas paralēlas svītras.



Minisumo robotu parametri — tie nosaka, kādam jābūt robotam, lai robots drīkstētu piedalīties sacensībās. Ja kāds no šiem parametriem netiek izpildīts, robots nedrīkst piedalīties sacensībās. Tie ir:

- Svars — 500 g;
- Izmērs 10 cm × 10 cm;
- Augstums neierobežots;
- Robots ir autonoms (sacensību laikā cilvēks neiejaucas robota darbībā);
- Robots sāk darboties piecas sekundes pēc palaišanas pogas nospiešanas vai pēc pults signāla.
- Robots nedrīkst bojāt ringu.
- Robots nedrīkst ietekmēt pretinieka darbību.
- No robota nedrīkst atdalīties detaļas.

Minisumo sacensību norise — sacensības dažādās pasaules valstīts ir ļoti līdzīgas ar nelielām izmaiņām. Sacensību norisi īsumā var atspoguļot ar šādām darbībām:

- Pēc tiesneša uzaicinājuma divi dalībnieki dodas pie sacensību laukuma ar saviem robotiem. (Lai netraucētu robotiem un neapdraudētu cilvēkus, skatītājiem jābūt tālāk no ringa).
- Pēc tiesneša signāla dalībnieki novieto savus minisumo robotus uz sacensību ringa tā, lai katrs atrastos savas līnijas pusē.
- Pēc starta signāla abi dalībnieki nospiež *Starta* pogas un atkāpjas no ringa.
- Pēc piecām sekundēm vai pēc tiesneša pults signāla minisumo roboti uzsāk cīņu.
- Zaudē tas robots, kurš pirmais tiek izstumts no ringa un pieskaras virsmai ārpus ringa.
- Ja robots neizkustas vai cīņa ievelkas, tiesnesis var apturēt cīņu un noteikt uzvarētāju.
- Šādas cīņas notiek trīs reizes, lai noskaidrotu labāko robotu.

Katrās sacensībās ir savi noteikumi, tāpēc ir svarīgi vienmēr iepazīties ar sacensību nolikumu un ievērot to!

8.1. DARBA LAPA. Minisumo uzstādījumi

Mērķis

Sagatavoties minisumo vadības programmas izstrādei.

Turpmākajās nodaļās, veidojot robota programmas kodu, tiks mainīts kods tikai galvenajā ciklā — funkcijā loop() un pievienoti daži mainīgie. Tāpēc tiks izmantoti iepriekšējā nodaļā izveidotie uzstādījumi funkcijā setup(), kas šeit netiks mainīti.

Uzdevumi

1. Uzrakstīt vai arī izmantot iepriekšējās nodaļas void setup() funkcijas sadaļu un motoru funkcijas.
2. Kompilēt un pārbaudīt programmas koda pareizību.

```
void setup() {   //Uzstādīšana
   //LED definīcijas
  pinMode(13,OUTPUT);   //Arduino LED
#define ledAON   digitalWrite(13,HIGH)
#define ledAOFF  digitalWrite(13,LOW)
  pinMode(5,OUTPUT);   //Left LED
#define led1ON   digitalWrite(5,HIGH);
#define led1OFF  digitalWrite(5,LOW);
  pinMode(11,OUTPUT);   //Right LED
#define led2ON   digitalWrite(11,HIGH);
#define led2OFF  digitalWrite(11,LOW);
  pinMode(2,OUTPUT);   //Back LED
#define led3ON   digitalWrite(2,HIGH);
#define led3OFF  digitalWrite(2,LOW);
   //Pogu definīcijas
  pinMode(12,INPUT);
  digitalWrite(12,HIGH);
#define button1 !digitalRead(12)
   //Seriālā porta monitora ieslēgšana
  Serial.begin(9600);
   //DIP slēdžu definīcija
  pinMode(15,INPUT);
  digitalWrite(15,HIGH);
#define DIP1 digitalRead(15)
  pinMode(14,INPUT);
  digitalWrite(14,HIGH);
#define DIP2 digitalRead(14)
  pinMode(16,INPUT);
  digitalWrite(16,HIGH);
#define DIP3 digitalRead(16)
  pinMode(17,INPUT);
  digitalWrite(17,HIGH);
#define DIP4 digitalRead(17)
   //Līnijas sensoru definīcija
  pinMode(A2,INPUT);
#define senLeft  analogRead(A2)<50
  pinMode(A1,INPUT);
#define senRight analogRead(A1)<50
  pinMode(A4,INPUT);
#define senBack  analogRead(A4)<50
   //SHARP sensoru definīcija
  pinMode(A5,INPUT);
#define SHARP1 !digitalRead(A5)
  pinMode(A3,INPUT);   //SHARP2
#define SHARP2 !digitalRead(A3)
  pinMode(7,INPUT);   //SHARP3
#define SHARP3 !digitalRead(7)
  pinMode(A8,INPUT);   //SHARP4
#define SHARP4 !digitalRead(A8)
  pinMode(A7,INPUT);   //SHARP5
#define SHARP5 !digitalRead(A7)
   //Motoru definīcija
  pinMode(3,OUTPUT);   //Kreisais PWM
  pinMode(4,OUTPUT);   //Kreisais Direction
  pinMode(9,OUTPUT);   //Labais PWM
  pinMode(10,OUTPUT);   //Labais Direction
}
   //Motoru vadības funkciju definīcijas
void Forward (int leftSpeed, int rightSpeed){
  analogWrite(3,leftSpeed);
  digitalWrite(4,HIGH);
  analogWrite(9,rightSpeed);
  digitalWrite(10,HIGH);
}
void Backward (int leftSpeed, int rightSpeed){
  analogWrite(3,leftSpeed);
  digitalWrite(4,LOW);
  analogWrite(9,rightSpeed);
  digitalWrite(10,LOW);
}
void Left (int leftSpeed, int rightSpeed){
  analogWrite(3,leftSpeed);
  digitalWrite(4,LOW);
  analogWrite(9,rightSpeed);
  digitalWrite(10,HIGH);
}
void Right (int leftSpeed, int rightSpeed){
  analogWrite(3,leftSpeed);
  digitalWrite(4,HIGH);
  analogWrite(9,rightSpeed);
  digitalWrite(10,LOW);
}

void loop() {   //Mūžīgais cikls
}
```

8.2. DARBA LAPA. Robots, kas izvairās no šķēršļiem

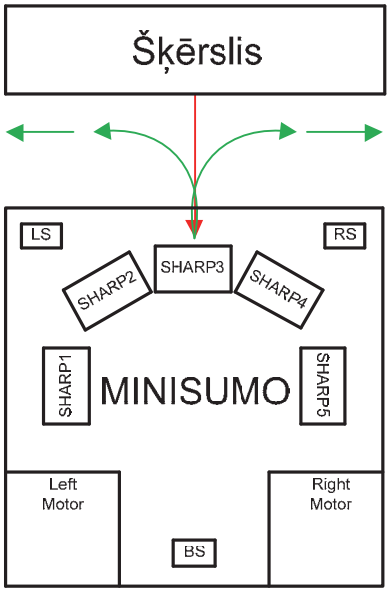
Mērķis

Izveidot robota vadības programmu, kas ļauj robotam izvairīties no priekšā esoša šķēršļa.

Uzdevumi

1. Izveidot robota darbības plānu.
   Ieraugot šķērsli ar priekšējo attāluma sensoru SHARP 3, liksim robotam pabraukt atpakaļ un pagriezties.

Robota darbību plāns, ieraugot šķērsli



2. Izveidot vadības programmu.
3. Pieslēgt robotu datora USB pieslēgvietai. Augšupielādēt programmu robotā.
4. Nolikt robotu brīvā vietā, kur tas nevar nokrist.
5. Ar spiedpogu iedarbināt robotu.
6. Ieraugot priekšā šķērsli, robots pabrauc atpakaļ un pagriežas pa kreisi. Pārbaudiet, vai tas tā notiek.
7. Atcerieties, ka `delay(200);` jālieto uzmanīgi, jo *delay* darbības laikā robots nereaģē uz sensoriem un akli pilda pēdējo darbību!

```
//Programmas sākums no 1. darba lapas

int motState =0;
   //Izveidojam mainīgo, kur norādīsim motora
   //darbību
void loop() {   //Mūžīgais cikls
  Forward(0,0);
    //Izsaucam funkciju motoru apstādināšanai
  if (button1)
    //Robots sāk darboties,  \nospiežot pogu 1
  {
    while (button1){
      //Cikls, kamēr poga ir nospiesta
    }
    while (!button1)
      //Robots apstājas līdz ar pogas 1
      //nospiešanu
    {
      if (SHARP3) {
        motState=1;
      }   //Ja priekšējais sensors noreaģē,
          //mainīgajā ieliek 1
      else {
        motState=0;
      }   //Ja priekšējais sensors nenoreaģē,
          //mainīgajā ieliek 0
        switch (motState){
           //Switch struktūra nolasa mainīgo
      case 0:
          //Ja mainīgais 0, robots brauc uz
                        priekšu
        Forward(150,150);
        break;
      case 1:
       //Ja mainīgais 1, robots brauc atpakaļ
        //un pagriežas
        Backward(150,150);
        delay(200);   //200 ms brauc atpakaļ
        Left(150,150);
        delay(200);   //200 ms pagriežas
        break;
      }
    }
    while (button1){
      //Cikls, kamēr poga 1 ir nospiesta
    }
  }
}
```

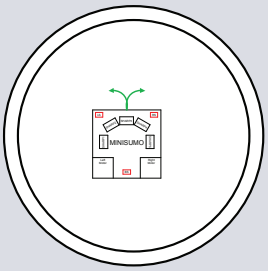
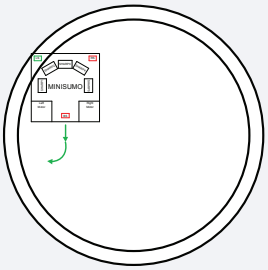
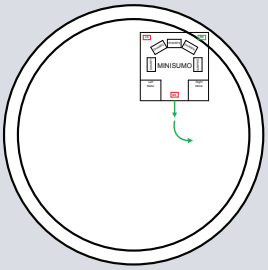
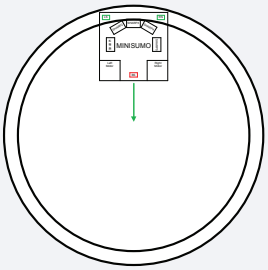
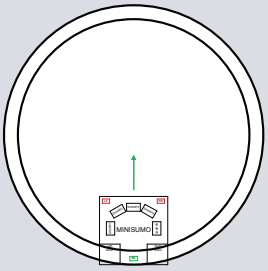
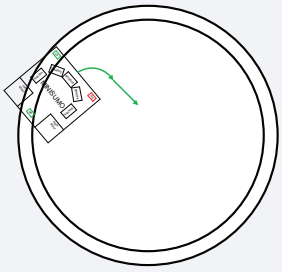
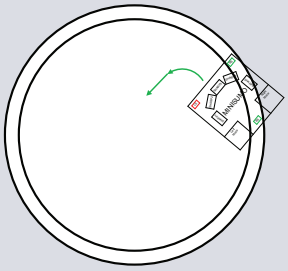
8.3. DARBA LAPA. Programma neizbraukšanai no ringa

Mērķis

Izveidot robota vadības programmu, kas neļauj tam izbraukt no sumo cīņas ringa.

Uzdevumi

1. Paņemiet robotu un, uzliekot uz ringa, pārbaudiet, cik dažādās kombinācijās var darboties līnijas sensori. Sensori nostrādā, "ieraugot" baltu līniju.
2. Padomājiet iespējamās stratēģijas, kā izvairīties no šādām situācijām.
3. Izveidot vadības programmu, ņemot vērā izpētītās kustības.
4. Pieslēgt robotu datora USB pieslēgvietai.
5. Augšupielādēt programmu robotā.
6. Nolieciet robotu uz laukuma.
7. Ar spiedpogu iedarbiniet robota programmu.
8. Novērot, kā robots darbojas laukumā un apstājas pie tā malas, pēc tam mainot braukšanas trajektoriju.
9. Ja robots neapstājas pie laukuma malas, tad ir jāpieregulē motora ātruma un laika aiztures "delay" vērtības.

Nr.	Attēls	senBack	senRight	senLeft	Darbības
0.		OFF	OFF	OFF	Pagriezties pa labi, apgriezties pa kreisi, braukt uz priekšu.
1.		OFF	OFF	ON	Braukt atpakaļ, atmuguriski pagriezties pa labi.
2.		OFF	ON	OFF	Braukt atpakaļ, atmuguriski pagriezties pa kreisi.
3.		OFF	ON	ON	Braukt atpakaļ.
4.		ON	OFF	OFF	Braukt uz priekšu.
5.		ON	OFF	ON	Pagriezties pa labi, braukt uz priekšu.
6.		ON	ON	OFF	Pagriezties pa kreisi, braukt uz priekšu.
7.	Nav iespējams uz trases	ON	ON	ON	Apstāties.

```
//Programmas sākums no iepriekšējās programmas

int motState =0;
   //Izveidojam mainīgo, kur norādīsim motora
   //darbību
void loop() {   //Mūžīgais cikls
  Forward(0,0);
    //Izsaucam funkciju motoru apstādināšanai
  if (button1)
    //Robots sāk darboties,  nospiežot pogu 1
  {
    while (button1){
      //Cikls, kamēr poga ir nospiesta
    }
    while (!button1)
      //Robots apstājas līdz ar pogas 1
      //nospiešanu
    {
      motState=0;   //Nonullē mainīgo
      if (senLeft) {
        led1ON;
        motState=motState+1;
      }
      else{
        led1OFF;
      }  //Pieskaitām motoru stavoklim sensora
         //bināro vērtību
      if (senRight) {
        led2ON;
        motState=motState+2;
      }
      else{
        led2OFF;
      }
      if (senBack) {
        led3ON;
        motState=motState+4;
      }
      else{
        led3OFF;
      }   //motState tagad glabā sensoru
          //stāvokļa vērtību
      switch (motState){
         //Switch struktūra nolasa mainīgo
      case 0:   //Neviens nav nostrādājis
        Forward(120,120);
        break;
      case 1:   //Kreisais sensors
        Backward(210,210);
        delay(100);
        Right(200,200);
        delay(150);
        break;
      case 2:   //Labais sensors
        Backward(210,210);
        delay(100);
        Left(200,200);
        delay(150);
        break;
      case 3:   //Abi priekšējie
        Backward(220,220);
        delay(250);
        break;
      case 4:   //Aizmugurējais
        Forward(255,255);
        delay(250);
        break;
      case 5:   //Aizmugurējais un kreisais
        Right (200,200);
        delay(100);
        Forward(150,150);
        delay(100);
        break;
      case 6:   //Aizmugurējais un labais
        Left (200,200);
        delay(100);
        Forward(150,150);
        delay(100);
        break;
      case 7:   //Visi sensori nostrādājuši
        Forward(0,0);
        break;
      }
    }
    while (button1){
      //Cikls, kamēr poga 1 ir nospiesta
    }
  }
}
```

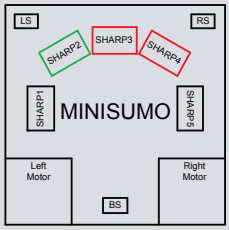
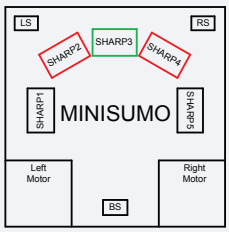
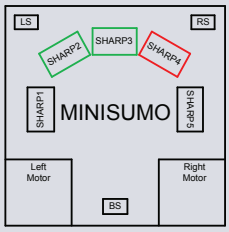
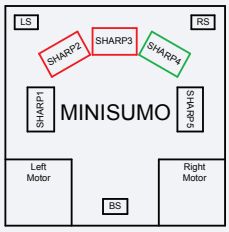
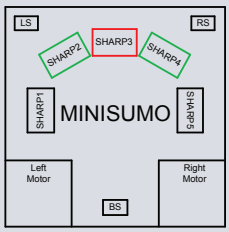
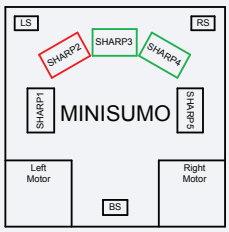
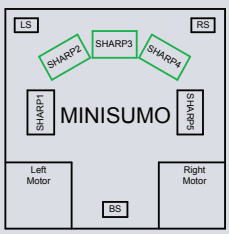
8.4. DARBA LAPA. Programma sumo cīņām

Mērķis

Izveidot robota cīņas programmu, izmantojot trīs priekšējos SHARP sensorus.

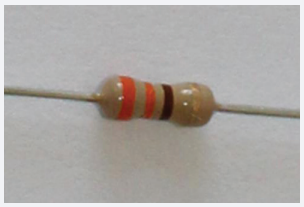
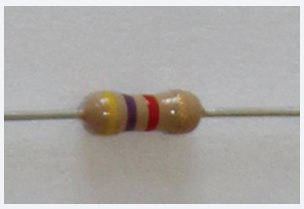
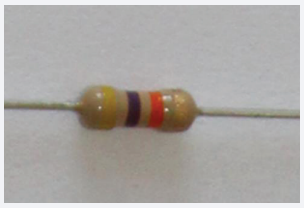
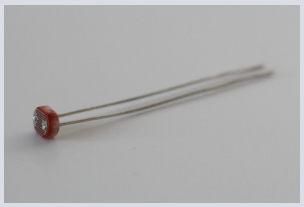
Uzdevumi

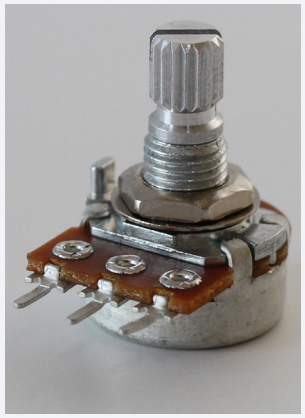
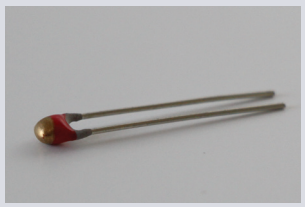
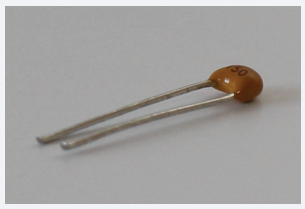
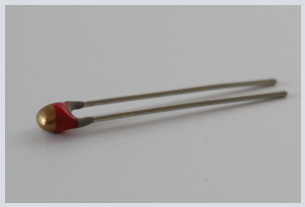
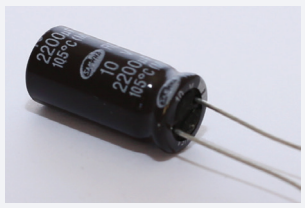
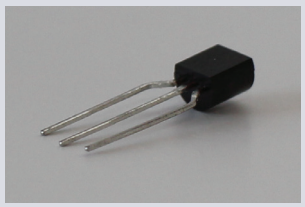
1. Ieslēdziet robotu.
2. Izmantojot gaismas diodes, kas iedegas pēc SHARP sensora nostrādes. Izdomājiet, kā būtu jārīkojas robotam, ar priekšējiem sensoriem ieraugot pretinieku.
3. Padomājiet iespējamās stratēģijas, kas ļautu veiksmīgi izstumt pretinieku no ringa, kad tas tiek pamanīts.
4. Izveidot vadības programmu, ņemot vērā izpētītās kustības.
5. Pieslēgt robotu datora USB pieslēgvietai.
6. Augšupielādēt programmu robotā.
7. Nolieciet robotu uz laukuma.
8. Ar spiedpogu iedarbiniet robota programmu.
9. Robotam vajadzētu sākt darboties pēc piecām sekundēm.
10. Ielieciet ringā vieglu kartona kasti vai otru robotu, un ļaujiet, lai robots atrod un izstumj to.
11. Papildiniet savu programmas kodu ar atlikušajiem SHARP sensoriem un izstrādājiet savu stratēģiju.
12. Pieregulējiet motoru ātruma vērtības.
13. Piedalieties minisumo sacensībās!

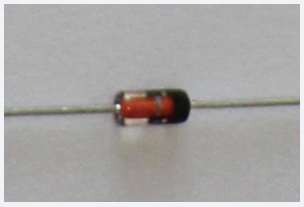
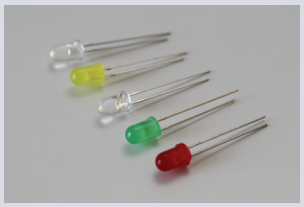
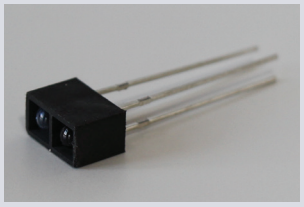
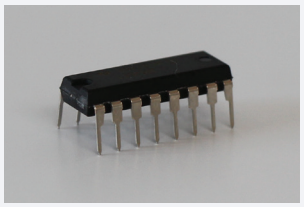
Nr.	Attēls	SHARP4	SHARP3	SHARP2	Darbības
8.		OFF	OFF	ON	Griežas pa kreisi.
16.		OFF	ON	OFF	Uz priekšu.
24.		OFF	ON	ON	Taisni un nedaudz pa kreisi.
32.		ON	OFF	OFF	Griežas pa labi.
40.		ON	OFF	ON	Pagriezties pa labi, braukt uz priekšu.
48.		ON	ON	OFF	Taisni un nedaudz pa labi.
56.		ON	ON	ON	Ātri braukt uz priekšu.

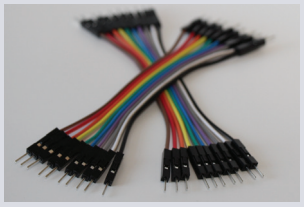
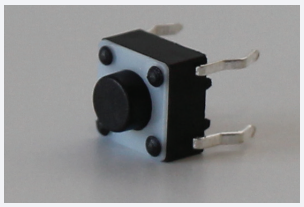
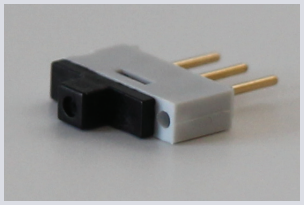
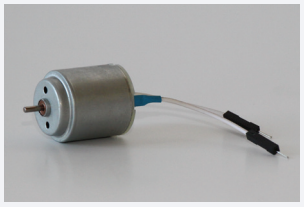
```
//Programmas sākums no iepriekš izveidotās
//programmas
int motState =0;
   //Izveidojam mainīgo, kur norādīsim motora
   //darbību
void loop() {   //Mūžīgais cikls
  Forward(0,0);
    //Izsaucam funkciju motoru apstādināšanai
  if (button1)
    //Robots sāk darboties, nospiežot pogu 1
  {
    while (button1){
      //Cikls, kamēr poga 1 ir nospiesta
    }
    led1ON;
    delay(1000);
    led2ON;
    delay(1000);
    led3ON;
    delay(1000);
    led1OFF;
    delay(1000);
    led2OFF;
    delay(1000);
    led3OFF;
    while (!button1)
      //Robots apstājas līdz ar pogas 1
      //nospiešanu
    {
      motState=0;   //Nonullē mainīgo
      if (senLeft) {
        motState=motState+1;
      }  //Pieskaitām motoru stāvoklim sensora
         //bināro vērtību
      if (senRight) {
        motState=motState+2;
      }
      if (senBack) {
        motState=motState+4;
      }
      if (motState==0){
        //SHARP sensorus saktisies tikai tad,
        //ja nav nostrādājis neviens linijas
        //sensors
        if (SHARP2) {
          motState=motState+8;
        }
        if (SHARP3) {
          motState=motState+16;
        }
        if (SHARP4) {
          motState=motState+32;
        }
      }   //motState tagad glabā sensoru
          //stāvokļa vērtību
      switch (motState){
        //Switch struktūra nolasa mainīgo
      case 0:   //Neviens nav nostrādājis
        Forward(120,120);
        break;
      case 1:   //Kreisais sensors
        Backward(210,210);
        delay(100);
        Right(200,200);
        delay(150);
        break;
      case 2:   //Labais sensors
        Backward(210,210);
        delay(100);
        Left(200,200);
        delay(150);
        break;
      case 3:   //Abi priekšējie
        Backward(220,220);
        delay(250);
        break;
      case 4:   //Aizmugurējais
        Forward(255,255);
        delay(250);
        break;
      case 5:   //Aizmugurējais un kreisais
        Right (200,200);
        delay(100);
        Forward(150,150);
        delay(100);
        break;
      case 6:   //Aizmugurējais un labais
        Left (200,200);
        delay(100);
        Forward(150,150);
        delay(100);
        break;
      case 7:   //Visi sensori nostrādājuši
        Forward(0,0);
        break;
           //Ja robotam ir nostrādājuši SHARP
           //sensori (priekšējie trīs)
      case 8:   //SHARP2
        Left(150,150);
        break;
      case 16:   //SHARP3
        Forward(200,200);
        break;
      case 24:   //SHARP2 & SHARP3
        Left(150,150);
        break;
      case 32:   //SHARP4
        Right(150,150);
        break;
      case 40:   //SHARP4 &SHARP2
        Forward(150,150);
        break;
      case 48:   //SHARP4 & SHARP3
        Right(150,150);
        break;
      case 56:   //SHARP2 &SHARP3 &SHARP4
        Forward(255,255);
        break;
      }
    }
    while (button1){
      //Cikls, kamēr poga 1 ir nospiesta
    }
  }
}
```

1. pielikums
Detaļu saraksts un attēli

Materiāls /detaļa	Izskats	Skaits
Rezistors (220 Ω)		1 gab.
Rezistors (330 Ω)		5 gab.
Rezistors (1 kΩ)		3 gab.
Rezistors (4,7 kΩ)		1 gab.
Rezistors (10 kΩ)		2 gab.
Rezistors (47 kΩ)		1 gab.
Fotorezistors (20 kΩ)		1 gab.

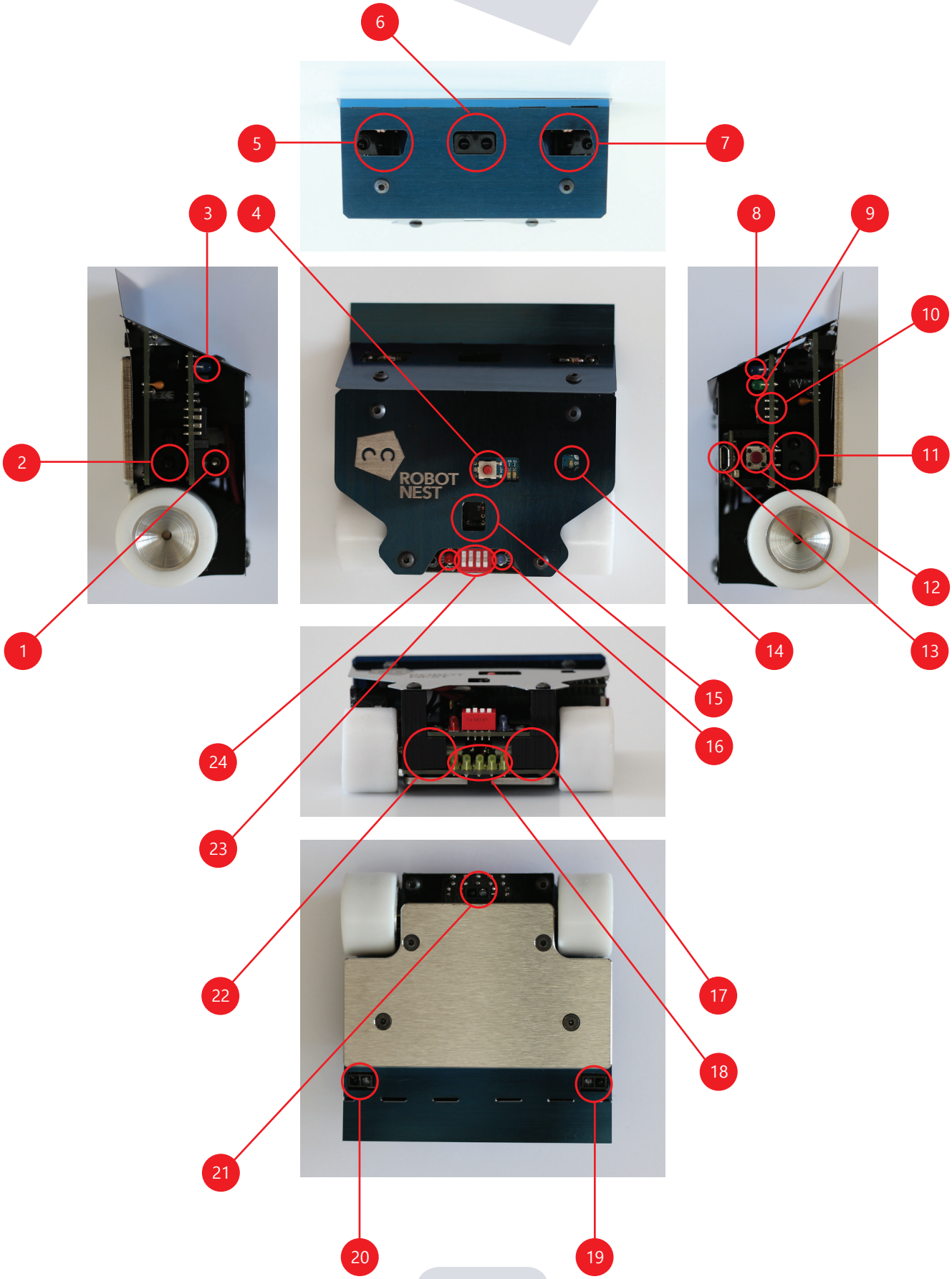
Materiāls /detaļa	Izskats	Skaits
Potenciometrs (50 kΩ)		3 gab.
Termorezistors NTC (2,2 kΩ)		1 gab.
Kondensators (15 pF)		1 gab.
Elektrolītiskais kondensators (100 µF, 63 V)		2 gab.
Elektrolītiskais kondensators (2200 µF, 10 V)		2 gab.
Tranzistors NPN BC517		2 gab.

Materiāls /detaļa	Izskats	Skaits
Diode PH4148		4 gab.
Mirdzdiode		5 gab.
RGB mirdzdiode		1 gab.
Optopāris ELITR9909		1 gab.
H tilta mikroshēma L293D		1 gab.

Materiāls /detaļa	Izskats	Skaits
Montāžas vadi		1 komplekts
Spiedpoga		2 gab.
Slēdzis		2 gab.
Patstāvīgā magnēta līdzstrāvas elektromotors (3–6 V)		1 gab.
Servomotors		1 gab.

2. pielikums

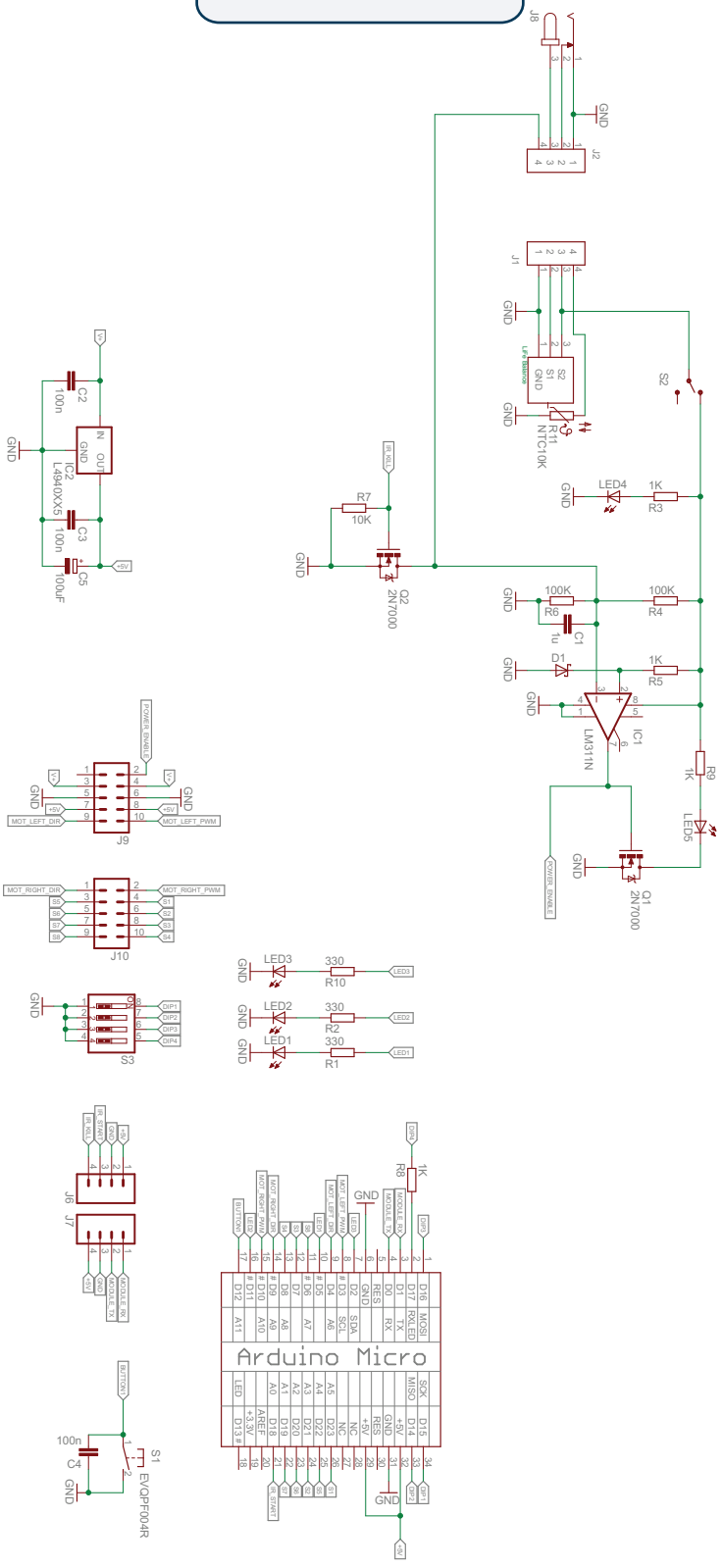
Robota ievadierīces un izvadierīces



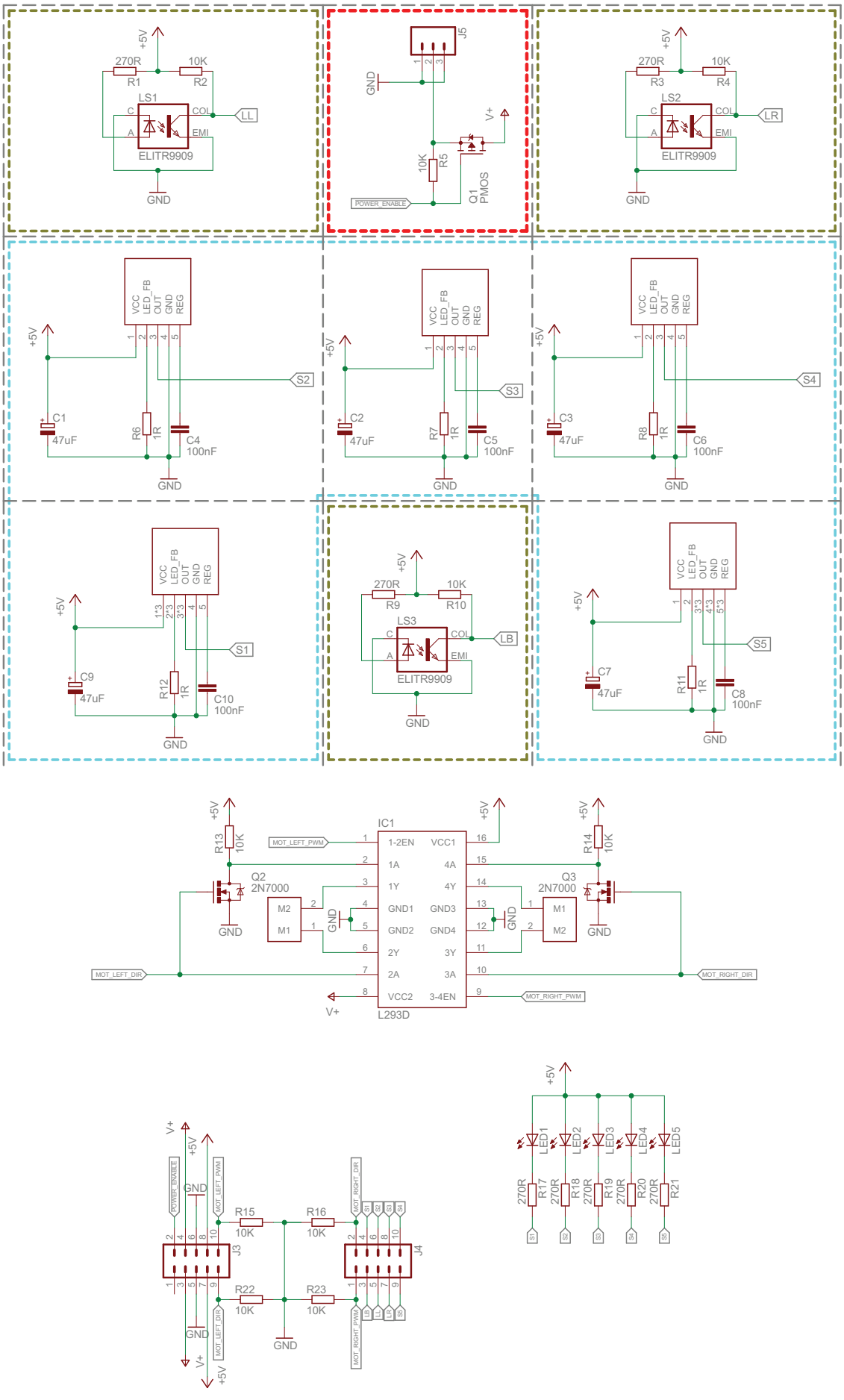
Nr.	Nosaukums	Pielietojums	Arduino kāja
1.	POWER SUPPLY	Pieslēgvieta lādētāja pieslēgšanai	–
2.	SHARP1	Pretinieka meklēšana	A5
3.	LED1	Programmējama diode informācijas izvadīšanai	5
4.	RESET	Poga programmas atstatīšanai	–
5.	SHARP2	Pretinieka meklēšana	A3
6.	SHARP3	Pretinieka meklēšana	7
7.	SHARP4	Pretinieka meklēšana	A8
8.	LED2	Programmējama diode informācijas izvadīšanai	11
9.	POWERLED	Norāda, kad robots ir ieslēgts un darbojas ar akumulatoru	–
10.	ON/OFF SWITCH	Akumulatora pieslēgšanas un atslēgšanas slēdzis	–
11.	SHARP5	Pretinieka meklēšana	A7
12.	BUTTON1	Programmējama poga	12
13.	MICRO USB	Arduino kontroliera USB pievads programmas augšupielādei, barošanai, seriālā komunikācija.	–
14.	LEDA	Programmējama diode informācijas izvadīšanai	13
15.	IR RECEVER	IR modulis robota iedarbināšanai un apstādināšanai ar pulti 18(HIGH) START	18
16.	LED3	Programmējama diode informācijas izvadīšanai	2
17.	MOTOR2	Vadāms 6 V līdzstrāvas motors. 9 — ātrums, 10 — virziens	9, 10
18.	SHARPLED	Iedegas, ja kaut kas nonāk SHARP redzes laukā. SHARP1 –SHARP5	–
19.	SENRIGHT	Optopāris minisumo laukuma malas noteikšanai	A1
20.	SENLEFT	Optopāris minisumo laukuma malas noteikšanai	A2
21.	SENBACK	Optopāris minisumo laukuma malas noteikšanai	A4
22.	MOTOR1	Vadāms 6 V līdzstrāvas motors. 3 — ātrums, 4 — virziens	3, 4
23.	DIPSWITCH1,2,3,4	Programmējams dip slēdzis robota stratēģijām DIP1(15), DIP2(14), DIP3(16), DIP4(17)	15, 14, 16, 17
24.	EMPTY BATERY	Diode, kas signalizē par zemu akumulatora līmeni	–

3. pielikums
Robota principiālā shēma

Augšējā iespiedplate



Apakšējā iespiedplate



4. pielikums

Mācies lodēt

P4.1. Mērķis

Temata mērķis ir iepazīstināt ar lodēšanas pamatiem un nepieciešamajiem piesardzības pasākumiem, lai lodēšana būtu droša un efektīva.

P4.2. Teorētiskā daļa

Ievads

Lodēšana ir viena no visnozīmīgākajām prasmēm elektronikas pasaulē. Elektronikas pamatus var apgūt arī bez prasmes lodēt, tomēr lodēšana paver plašākas iespējas īstenot interesantus projektus, kā arī pievienoties plašajam elektronikas entuziastu pulkam. Pasaulē arvien vairāk tiek izmantotas dažādas elektroniskas un elektrotehniskas iekārtas, tādēļ šī prasme ir ļoti nozīmīga. Būtiska elektronikas entuziasta ikdienas sastāvdaļa ir ne tikai spēja saprast, bet arī prasme būvēt, labot un papildināt savas elektroniskās iekārtas.

Šajā nodaļā tiks aplūkoti lodēšanas pamati, koncentrējoties uz tādu detaļu lodēšanu, kas tiek montētas montāžas platē (angļu val. — *through-hole soldering* vai *plated trhough-hole soldering* — *PTH*). Tiks sniegts ievads par lietotajiem instrumentiem un materiāliem, kā arī nedaudz aplūkota jau samontētu plašu labošana.

Izmantotie materiāli

Galvenais lodēšanas materiāls ir lodalva. Kā var noprast no šī materiāla nosaukuma, tas ir dažādu mīkstu metālu un palīgmateriālu savienojums, kas parasti ir līdzīgs drātij, ir satīts spolē vai citos ērti lietojamos iesaiņojumos.

Atkarībā no veicamā lodēšanas uzdevuma jāizvēlas pēc ķīmiskā sastāva un resnuma atbilstoša lodalva. Neiedziļinoties sīkumos, lodalva pēc tās ķīmiskā sastāva tiek iedalīta: **svinu saturoša** un **svinu nesaturoša**. Vēsturiski svinu (*Pb*) izmanto kombinācijā ar alvu (*Sn*), lai nodrošinātu zemāku lodalvas kušanas temperatūru un labāku plūšanu, kas ir būtiska laba kontakta nodrošināšanai starp detaļām.

Rūpējoties par dabas aizsardzību un cilvēku veselību, no 2006. gada vairākas pasaules valstis ir pilnībā aizliegušas elektronikā izmantot svina lodalvu. Nonākot lielā daudzumā cilvēka organismā, svins uzkrājas un ar laiku var izraisīt saindēšanos. Tāpēc vienmēr svarīgi atcerēties — pabeidzot darbu ar svina lodalvu, vienmēr **rūpīgi jānomazgā rokas**.

Lai izvairītos no svina nonākšanas organismā, var izmantot svinu nesaturošu lodalvu. Tomēr jāņem vērā, ka tās kušanas temperatūra ir augstāka, kā arī saķere ar citiem materiāliem zemāka. Saķeres uzlabošanai tiek izmantoti daudzveidīgi kušņi (angļu val. — *flux*). Ir lodalvas, kuru serdē ir kušņi, tādēļ tos var neiegādāties atsevišķi.

Lodalvas iesaiņojumi

Lodalvas resnums ir atkarīgs no detaļu izmēra, kuras vēlaties salodēt. Jo lielāka detaļa, jo resnāka lodalva. Šeit veicamajiem darbiem jums ērtākas būs tādas lodalvas, kuru diametrs nepārsniedz 1,0 mm.

Izmantotie instrumenti

Lodēšanai bez minētajiem materiāliem nepieciešams lodāmurs, lodāmura statīvs, kā arī dažādi palīglīdzekļi lodalvas noņemšanai un detaļu pieturēšanai.

Lodāmurs ir elektriskais sildītājs, kas uzsilda lodalvu līdz tās kušanas temperatūrai. Tādēļ kā jebkurai elektriskai iekārtai arī šai ir jāievēro tās lietošanas instrukcija, kuru izstrādājis konkrētā lodāmura ražotājs. Specifisku uzdevumu veikšanai eksistē karstā gaisa un gāzes lodāmuri, bet šeit tiks aplūkoti tikai elektriskie.

Lodāmuri ir ārkārtīgi daudzveidīgi, lai ar tiem efektīvi varētu paveikt nepieciešamo uzdevumu. Tiem ir dažādi uzgaļi, dažādas elektriskās jaudas, dažādas temperatūras un regulēšanas iespējas. Tomēr galvenais rādītājs ir lietošanas ērtība. Tādēļ izvēlieties tādu lodāmuru, kas jums šķiet ērts. Ja izvēlēsieties pārāk lielu, tad būs grūti salodēt mazas detaļas. Ja tas būs pārāk mazs, tad visticamāk nevarēsiet pietiekami sasildīt detaļas un lodalva nepielips pie tām. Ieteicams lietot lodāmuru ar konisku uzgali, jo tas iesācējam būs vienkāršāks.

Lodāmurs ar konisku uzgali

Papildu lodāmuram dažādu darbu veikšanai ļoti noderīgi ir:

Vara izlodēšanas lente — ļauj ērti noņemt lieko lodalvu no montāžas plates vai detaļām, jo izkusušo lodalvu burtiski iesūc sevī. Tā sastāv no smalkām savītām vara stieplēm.

Vara izlodēšanas lente

Alvas atsūcējs — ļauj ērti noņemt lieko lodalvu, izmantojot vakuumu.

Alvas atsūcējs

Trešā roka — detaļu un citu palīglīdzekļu turētājs, kas ir īpaši noderīgs, ja neesat pieredzējis lodētājs.

Trešā roka

Lodēšana

Ir grūti lodēšanas procesu paskaidrot rakstiskā formā, jo tas saistīts ar cilvēka iemaņām un uzmanību. Tomēr ir būtiski ievērot dažus priekšnosacījumus labam lodējumam:

1. Esiet uzmanīgs ar karstu lodāmuru.
2. Darba vietu vienmēr turiet tīru; vēlams neieturēt maltīti darba vietā (atcerieties par svina kaitīgumu).
3. Lietojiet trešo roku, kad vien tā var noderēt.
4. Ja iespējams, uzturiet lodāmura temperatūru ap 350 °C.
5. Ja pamanāt dūmus no lodāmura, samaziniet tā temperatūru vai atslēdziet to pavisam.
6. Izmantojiet lodāmura tīrītāju (mitrs sūklis vai speciāla uzgaļa tīrāmā pasta) pirms katra lodējuma.
7. Lodēšanai izmantojiet lodāmura uzgaļa sānus, nevis tā pašu galu.
8. Lai nodrošinātu labu kontaktu, mēģiniet vienlaicīgi uzsildīt abas detaļas, kas tiek lodētas.
9. Kad detaļa pielodēta, vispirms noņemiet lodalvas stiepli un tikai pēc tam lodāmuru.
10. Labs detaļas lodējums pie montāžas plates līdzinās vulkāna smailei, nevis bumbai vai lodalvas nekārtīgai kaudzei.

Lai saprastu, kāda ir pareiza un nepareiza lodēšanas tehnika, izpētiet attēlu "Lodēšanas tehnika".

P4.1. DARBA LAPA

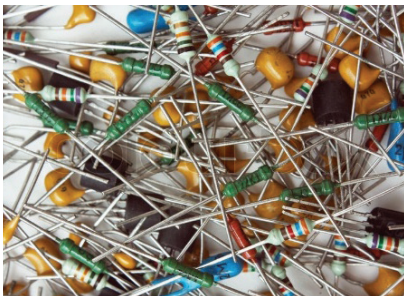
Mērķis

Salodēt savu pirmo detaļu pasniedzēja vadībā.

Nepieciešamie materiāli

Lodāmurs un lodalva ar kusni vai kusnis atsevišķi.

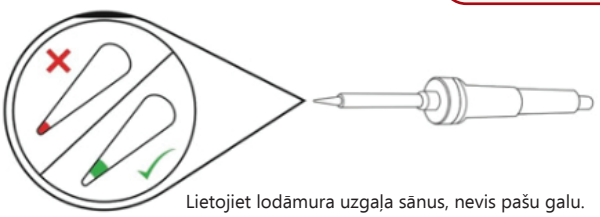
Materiāli



Darba izpildes soļi

1. Sagatavojiet darba vietu tā, lai uzkarsis lodāmurs neradītu apdeguma briesmas, un brīvās vietas būtu pietiekami daudz, lai varētu darboties ar abām rokām.
2. Sagatavojiet lodējamos materiālus — nelielus vada gabaliņus, vecas elektronikas detaļas vai kādus citus lodējamus materiālus, kurus jums izsniedzis pasniedzējs.
3. Attīriet detaļas no oksīda kārtiņas. To var darīt ar kušņa palīdzību (ja tas ir atsevišķā iesaiņojumā) vai abrazīvu materiālu, piemēram, smalku smilšpapīru. Detaļas nav jāattīra, ja jūsu lietotās lodalvas serdē ir kusnis.
4. Lodējiet atbilstoši pasniedzēja norādījumiem.
5. Ļaujiet detaļām atdzist un pārbaudiet, vai lodējums ir izdevies atbilstoši pasniedzēja norādēm.

Lodēšanas tehnika



Lietojiet lodāmura uzgaļa sānus, nevis pašu galu.

Sasildiet detaļu un montāžas plati vienlaicīgi.

Kamēr turiet lodāmuru saskarē ar abām detaļām, uzmanīgi un vienmērīgi spiediet lodalvas stiepli pret tām.

Nemēģiniet lodalvu "pielipināt" lodāmura uzgalim un tad to pievienot detaļām.

Vienmēr, kad uz uzgaļa veidojas melns apdegums, notīriet to ar mitru sūkli vai lodāmura ražotāja piedāvāto tīrīšanas līdzekli.

A Izkususī lodalva vienmērīgi piepilda montāžas plates caurumu un pielīp detaļai, ieņemot vulkāna virsotnei līdzīgu formu.

B Ja lodalva nepielīp montāžas platei, bet pieraujas detaļai, izmantojot kusni.

C Ja lodējums nelīdzinās vulkānam, pievienojiet kusni un pēc tam lodalvu.

D Ja lodējums ir neglīts, pievienojiet kusni un pēc tam lodalvu, kamēr izveidojas vulkānam līdzīga virsma.

E Ja lodalvas par daudz, tad izmantojiet kādu no lodalvas noņemšanas palīglīdzekļiem.